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

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: 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.

1.3.2 Abreviações

RF: Requisito Funcional.

RNF: Requisito Não Funcional.

Pr.: 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 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 a um gerente 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 liquida 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.
Ator 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.
Ator 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.
Ator 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 public class Model<T> {
21     protected Integer id;
22     //transient para a serializacao nao incluir.
23     protected transient Context context;
24
25     public Model(Context context){
26         this.context = context;
27     }
28
29     public void setContext(Context context){
30         this.context = context;
31     }
32
33     public void save(){
34         save(null);
35     }
36
37     public void save(final RequestCallback<Model> callback){
38         new ServerRequest(context).sendRequest(getControllerName(), ServerRequest.Action.SAVE, new Gson().toJson(this),
       new RequestCallback<JSONObject>() {
39             @Override
40             public void execute(JSONObject json) throws Exception {
41                 setId(json.getInt("id"));
42                 if (callback != null){
43                     callback.execute(Model.this);
44                 }
45
46                 super.execute(json);
47             }
48         });
49     }
50
51     public void delete(){
52         new ServerRequest(context).sendRequest(getControllerName(), ServerRequest.Action.DELETE, new Gson().toJson(this
       ), null);
53     }
54
55     public static <T extends Model> void find(Context context, Class<T> klass, final Type typeArray, final
       RequestCallback<T> callback, int id){
56         find(context, klass, typeArray, callback, id + "");
57     }
58
59     public static <T extends Model> void find(final Context context, Class<T> klass, final Type typeArray, final
       RequestCallback<T> callback, String where){
60         ServerRequest serverRequest = new ServerRequest(context);
61         String controller = ((Class<T>) ((ParameterizedType) klass.getGenericSuperclass()).getActualTypeArguments()[0])
       .getSimpleName();
62         serverRequest.sendRequest(controller, ServerRequest.Action.FIND, where, new RequestCallback<JSONObject>() {
63             @Override
64             public void execute(JSONObject json) throws Exception {
65                 Log.w(Utils.TAG, "Trying cast to " + typeArray.toString());
66
67                 ArrayList<T> lista = new Gson().fromJson(json.getJSONArray("data").toString(), typeArray);
68                 for (T item : lista){
69                     item.setContext(context);
70                 }
71                 if (callback != null){
72                     if (lista.size() > 0) {
73                         callback.execute(lista.get(0));
74                     }
75                     callback.execute(lista);
76                 }
77                 super.execute(json);
78             }
```

```
        });
    }

//    public static <T extends Model> void find(Context context, Class<T> klass, RequestCallback callback){
//        find(context, klass, callback, "");
//    }

    public Integer getId() {
        return id;
    }

    public T setId(Integer id) {
        this.id = id;
        return (T) this;
    }

    protected String getControllerName(){
        return this.getClass().getSimpleName();
    }
}
```

Código 1: Model.java

4.1.2 Comanda

```
package ds2.equipe1.restaurante.modelos;

import android.content.Context;

import java.util.ArrayList;
import java.util.Date;

public class Comanda extends Model<Comanda> {

    private ArrayList<Pedido> pedidos = new ArrayList<>();
    private String data;
    private boolean ativa = false;
    private String nome;

    public Comanda(Context context, String nome, Pedido primeiro){
        super(context);
        Date agora = new Date();
        pedidos.add(primeiro);
        data = agora.toString();
        this.nome = nome;
        ativa = true;
    }

    public void addPedido(Pedido outro){
        pedidos.add(outro);
    }

    public void setNome(String nome){
        this.nome=nome;
    }

    public ArrayList< Pedido > getPedidos(){
        return pedidos;
    }

    public Pedido getPedido(int indice){
        return pedidos.get(indice);
    }

    public void removerPedido(Pedido pedido){
        pedidos.remove(pedido);
    }

    public boolean estaAtiva(){
        return ativa;
    }

    public void desativar(){
        ativa = false;
    }

    public float getCustoTotal(){
        float custo = 0;
        for (Pedido pedido : pedidos){
            custo+= pedido.getCusto();
        };
        return custo;
    }
}
```

Código 2: Comanda.java

4.1.3 Compra

```
1  package ds2.equipe1.restaurante.modelos;
2
3  import android.content.Context;
4
5  public class Compra extends Model<Compra> {
6      private Item item;
7      private int quantidade;
8      private float preco;
9      private String data;
10     private Fornecedor fornecedor;
11
12     public Compra(Context context){
13         super(context);
14     }
15
16     public Compra(Context context, Item item, int quantidade, float preco, String data, Fornecedor fornecedor) {
17         super(context);
18         this.item = item;
19         this.quantidade = quantidade;
20         this.preco = preco;
21         this.data = data;
22         this.fornecedor = fornecedor;
23     }
24
25     public String getNomeDoItem(){
26         return this.item.getNome();
27     }
28
29     public int getQuantidade(){
30         return this.quantidade;
31     }
32
33     public float getPreco() {
34         return this.preco;
35     }
36
37     public String getData() {
38         return this.data;
39     }
40
41     public Fornecedor getFornecedor() { return this.fornecedor; }
42 }
```

Código 3: Compra.java

4.1.4 Endereço

```
1  package ds2.equipe1.restaurante.modelos;
2
3  import android.content.Context;
4
5  public class Endereco extends Model<Endereco> {
6      private String logradouro;
7      private String rua;
8      private int numero;
9      private String bairro;
10     private String cidade;
11     private String estado;
12     private String cep;
13
14     public Endereco(Context context, String logradouro, String rua, int numero, String bairro, String cidade, String
        estado, String cep) {
15         super(context);
16         this.logradouro = logradouro;
17         this.rua = rua;
18         this.numero = numero;
19         this.bairro = bairro;
20         this.cidade = cidade;
21         this.cep = cep;
22         this.estado = estado;
23     }
24
25     public String getLogradouro() {
26         return logradouro;
27     }
28
29     public String getRua() {
30         return rua;
31     }
32
33     public int getNumero() {
34         return numero;
35     }
36
37     public String getBairro() {
```

```
        return bairro;
    }

    public String getCidade() {
        return cidade;
    }

    public String getCep() {
        return cep;
    }

    public String getEstado() {
        return estado;
    }

    public void setEstado(String estado) {
        this.estado = estado;
    }
}
```

Código 4: Endereco.jav

4.1.5 Fornecedor

```
package ds2.equipe1.restaurante.modelos;

import android.content.Context;

public class Fornecedor extends Model<Fornecedor> {
    private String nome;
    private String telefone;
    private String cnpj;
    private String email;
    private Endereco endereco;

    public Fornecedor(Context context){
        super(context);
    }

    public Fornecedor(Context context, String nome, String telefone, String cnpj, String email){
        super(context);
        this.nome = nome;
        this.telefone = telefone;
        this.cnpj = cnpj;
        this.email = email;
    }

    public String getNome(){
        return nome;
    }

    public String getTelefone (){
        return telefone;
    }

    public String getCnpj(){
        return cnpj;
    }

    public String getEmail(){
        return email;
    }

    public void setNome(String nome){
        this.nome = nome;
    }

    public void setTelefone (String telefone){
        this.telefone = telefone;
    }

    public void setCnpj(String cnpj){
        this.cnpj = cnpj;
    }

    public void setEmail(String email){
        this.email = email;
    }

    public Endereco getEndereco() {
        return endereco;
    }

    public void setEndereco(Endereco endereco) {
        this.endereco = endereco;
    }
}
```

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  public class Funcionario extends Model<Funcionario> {
7      private String nome;
8      private Integer id_endereco;
9      private String telefone;
10     private String cpf;
11     private String nome_de_usuario;
12     private Integer tipo;
13
14     public Funcionario(Context context){
15         super(context);
16     }
17
18     public Funcionario(Context context, String nome, Integer id_endereco, String telefone, String cpf, String
        nome_de_usuario, Integer tipo) {
19         super(context);
20         this.nome = nome;
21         this.id_endereco = id_endereco;
22         this.telefone = telefone;
23         this.cpf = cpf;
24         this.nome_de_usuario = nome_de_usuario;
25         this.tipo = tipo;
26     }
27
28     public String getNome() {
29         return nome;
30     }
31
32     public Integer getIdEndereco() {
33         return id_endereco;
34     }
35
36     public String getTelefone() {
37         return telefone;
38     }
39
40     public String getCpf() {
41         return cpf;
42     }
43
44     public String getNome_de_usuario() {
45         return nome_de_usuario;
46     }
47
48     public Integer getTipo () { return tipo; };
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
68     public void setTipo (Integer tipo) {this.tipo = tipo; };
69 }
```

Código 6: Funcionario.java

4.1.7 Garçom

```java
package ds2.equipe1.restaurante.modelos;

import android.content.Context;

public class Garcom extends Funcionario {
    public Garcom(Context context, String nome, Integer id_endereco, String telefone, String cpf, String
     nome_de_usuario, Integer tipo) {
        super(context, nome, id_endereco, telefone, cpf, nome_de_usuario, tipo);
    }
}
```

Código 7: Garcom.java

4.1.8 Gerente

```java
package ds2.equipe1.restaurante.modelos;

import android.content.Context;

public class Gerente extends Funcionario {
    public Gerente(Context context, String nome, Integer id_endereco, String telefone, String cpf, String
     nome_de_usuario, Integer tipo) {
        super(context, nome, id_endereco, telefone, cpf, nome_de_usuario, tipo);
    }
}
```

Código 8: Gerente.java

4.1.9 Ingrediente

```java
package ds2.equipe1.restaurante.modelos;

import android.content.Context;

public class Ingrediente extends Model<Ingrediente> {
    private Item item;
    private int quantidade;

    public Ingrediente(Context context, Item item, int quantidade) {
        super(context);
        this.item = item;
        this.quantidade = quantidade;
    }

    public Item getItem() {
        return item;
    }

    public int getQuantidade() {
        return quantidade;
    }
}
```

Código 9: Ingrediente.jav

4.1.10 Item

```java
package ds2.equipe1.restaurante.modelos;

import android.content.Context;

import ds2.equipe1.restaurante.controles.ControleDeAtendimento;

public class Item extends Model<Item> {
    private String nome;
    private String unidade;
    private int quantidade;
    private int limiteMinimo;


    public Item(Context context){
        super(context);
    }

    public Item(Context context, String nome, int quantidade, String unidade, int limiteMinimo){
        super(context);
```

```
20          this.nome = nome;
21          this.unidade = unidade;
22          this.quantidade = quantidade;
23          this.limiteMinimo = limiteMinimo;
24      }
25
26      public void verificarItemAbaixoDoLimite(int quantidadeParaReduzir){
27          if (getQuantidade()-quantidadeParaReduzir < getLimiteMinimo()){
28              new Aviso(context, this).save();
29          }
30      }
31
32      public String getNome(){
33          return nome;
34      }
35
36      public int getQuantidade(){
37          return quantidade;
38      }
39
40      public int getLimiteMinimo(){
41          return limiteMinimo;
42      }
43
44      public void setQuantidade(int quantidade){
45          //alterar quantidade no banco
46          this.quantidade = quantidade;
47      }
48
49      public void setLimiteMinimo(int limiteMinimo){
50          //alterar limiteMinimo no banco
51          this.limiteMinimo = limiteMinimo;
52      }
53
54      public void setNome(String nome){
55          this.nome = nome;
56      }
57
58      public void setUnidade(String unidade){
59          this.unidade=unidade;
60      }
61  }
```

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   public class Mesa extends Model<Mesa> {
8       private int numero;
9       private ArrayList<Comanda> comandas = new ArrayList< Comanda >();
10
11      public Mesa(Context context, int numero){
12          super(context);
13          //inserir chamada do SQL para carregar comandas ativas
14          this.numero = numero;
15      }
16
17      public void addComanda(Comanda nova){
18          comandas.add(nova);
19      }
20
21      public ArrayList< Comanda > getComandas(){
22          return comandas;
23      }
24
25      public ArrayList < Comanda > getComandasAtivas(){
26          ArrayList < Comanda > ativas = new ArrayList < Comanda >();
27          for (int i = 0; i < comandas.size(); i++) {
28              Comanda atual = comandas.get(i);
29              if(atual.estaAtiva()){
30                  ativas.add(atual);
31              }
32          }
33          return ativas;
34      }
35
36      public Comanda getComanda(int indice){
37          return comandas.get(indice);
38      }
39  }
```

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  public class Pedido extends Model<Pedido> {
8
9      int quantidadeDeProdutos;
10     boolean entregue;
11     ArrayList <Produto> produtos = new  ArrayList <Produto>();
12
13     public Pedido(Context context, ArrayList<Produto> produtos){
14         super(context);
15         this.produtos = produtos;
16         quantidadeDeProdutos = produtos.size();
17         entregue = false;
18     }
19
20     public float getCusto(){
21         float custo = 0;
22         for (int i =0;i< produtos.size();i++) {
23             custo += produtos.get(i).getPreco();
24         }
25         return custo;
26     }
27
28     public void setEntregue(){
29         entregue = true;
30     }
31
32     public int getQuantidadeDeProdutos(){
33         return quantidadeDeProdutos;
34     }
35
36     public ArrayList <Produto> getProdutos(){
37         return produtos;
38     }
39
40     public Produto getProduto(int indice){
41         return produtos.get(indice);
42     }
43 }
```

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  public class Produto extends Model<Produto> {
8      private String nome;
9      private float preco;
10     private ArrayList<Ingrediente> ingredientes;
11
12     public Produto(Context context, String nome, float preco, ArrayList<Ingrediente> ingredientes) {
13         super(context);
14         this.nome = nome;
15         this.preco = preco;
16         this.ingredientes = ingredientes;
17     }
18
19     /*public static ArrayList<Produto> carregarProdutos() {
20         ArrayList<Produto> listaDeProdutos = new ArrayList<Produto>();
21
22         ArrayList<Produto> lista = new ArrayList<Produto>();
23
24         //conexao com o banco estabelecida
25         for (int i = 0; i < lista.count(); i++) {
26             listaDeProdutos.add(i, new Produto(lista.nome, lista.preco, lista.id, ...));
27         }
28
29         return listaDeProdutos; //(???)
30     }*/
31
32     public boolean validarProduto() {
33         /*for (int i = 0; i < ingredientes.size(); i++) {
34             if (ingredientes.get(i).getQuantidade() > ControleDeItem.consultarItem(ingredientes.get(i).getItem().
       getNome(), new RequestCallback<Item>() {
35                 @Override
36                 public void execute(ArrayList<Item> itens) {
```

```
                    BuscaItem.this.itens.clear();
                    BuscaItem.this.itens.addAll(itens);
                    adapter.notifyDataSetChanged();
                    super.execute(itens);
                }
            }).getQuantidade()) {
                return false;
            }
        }*/
        return true;
    }

    //Deve ser chamado quando o garcom conclui o pedido.
    public void alertarSobreItensAbaixoDoLimite(){
        for (Ingrediente ingrediente : getIngredientes()){
            Item item = ingrediente.getItem();
            item.verificarItemAbaixoDoLimite(ingrediente.getQuantidade());
        }
    }

    public String getNome(){
        return nome;
    }

    public float getPreco(){
        return preco;
    }

    public ArrayList <Ingrediente> getIngredientes(){
        return ingredientes;
    }

    public void setPreco(float preco){
        this.preco = preco;
    }
}
```

Código 13: Produto.java

4.2 Controles

4.2.1 Controle de Atendimento

```
package ds2.equipe1.restaurante.controles;

import android.content.Context;

import java.util.ArrayList;

import ds2.equipe1.restaurante.modelos.Comanda;
import ds2.equipe1.restaurante.modelos.Mesa;
import ds2.equipe1.restaurante.modelos.Pedido;
import ds2.equipe1.restaurante.modelos.Produto;

public class ControleDeAtendimento {
    private Context context;
    private ControleDeImpressao controleDeImpressao;

    private ArrayList<Mesa> mesas = new ArrayList<Mesa>();

    public ControleDeAtendimento(Context context){
        this.context = context;
        this.controleDeImpressao = new ControleDeImpressao(context);

        //inserir comando sql pra saber numero de mesas
        int numeroDeMesas = 20; //puxar numero do sql
        for(int i = 1; i <= numeroDeMesas; i++) {
            mesas.add(new Mesa(context, i));
        }
    }

    public void criarComanda(Mesa mesa, Pedido pedido, String nome){
        mesa.addComanda(new Comanda(context, nome, pedido));
    }

    public void encerrarComanda(Comanda comanda){
        comanda.desativar();
    }

    public void imprimirPedido(Pedido pedido){
        ArrayList<Produto> listaDeProdutos = pedido.getProdutos();
        for(int i = 0; i < listaDeProdutos.size(); i++){
            controleDeImpressao.imprimir(listaDeProdutos.get(i).getNome());
        }
    }

```

```
44      public void imprimirConta(Comanda comanda){
45          String saida = "";
46          ArrayList < Pedido > listaDePedidos = comanda.getPedidos();
47          for(int i = 0; i < listaDePedidos.size(); i++) {
48              ArrayList < Produto > listaDeProdutos = listaDePedidos.get(i).getProdutos();
49              for(int j = 0; j < listaDeProdutos.size(); j++) {
50                  saida += listaDeProdutos.get(i).getNome() + "␣" + listaDeProdutos.get(i).getPreco() + "\n";
51              }
52          }
53          float total = comanda.getCustoTotal();
54          saida += "Total␣de␣Pedidos" + total + "\n␣10%␣=␣" + 0.1*total + "\n␣Total␣" + 1.1*total;
55
56          controleDeImpressao.imprimir(saida);
57      }
58
59      public ArrayList < Produto > consultarPedido(Pedido pedido){
60          return pedido.getProdutos();
61      }
62
63      public void incluirPedido(Comanda comanda, Pedido pedido){
64          comanda.addPedido(pedido);
65      }
66
67      public void entregarPedido(Pedido pedido){
68          pedido.setEntregue();
69      }
70
71      public void cancelarPedido(Comanda comanda, Pedido pedido) {
72          comanda.removerPedido(pedido);
73      }
74
75      public ArrayList < Produto > consultarCardapio() {
76          ArrayList < Produto > cardapio = new ArrayList < Produto >();
77          //Aqui so com consulta ao DB ne? pq precisa pegar a lista de produtos.
78          //fica pra outra hora beijo no coracao!
79          return cardapio;
80      }
81
82      public ArrayList < Comanda > relatorioComandasAtivas(){
83          ArrayList < Comanda > ativas = new ArrayList < Comanda >();
84          //preciso de consulta ao banco neste ponto
85          return ativas;
86      }
87
88
89      public ArrayList < Comanda > relatorioComandasEncerradas(){
90          ArrayList < Comanda > encerradas = new ArrayList < Comanda >();
91          //preciso de consulta ao banco neste ponto
92          return encerradas;
93      }
94
95      public void relatorioArrecadacaoBruta(){
96          //preciso de consulta ao banco neste ponto
97      }
98
99  }
```

Código 14: ControleDeAtendimento.java

4.2.2 Controle de Compra

```
1   package ds2.equipe1.restaurante.controles;
2
3   import android.content.Context;
4   import android.util.Log;
5
6   import com.google.gson.reflect.TypeToken;
7
8   import java.util.ArrayList;
9
10  import ds2.equipe1.restaurante.helpers.RequestCallback;
11  import ds2.equipe1.restaurante.helpers.Utils;
12  import ds2.equipe1.restaurante.modelos.Compra;
13  import ds2.equipe1.restaurante.modelos.Fornecedor;
14  import ds2.equipe1.restaurante.modelos.Item;
15  import ds2.equipe1.restaurante.modelos.Model;
16
17  public class ControleDeCompra {
18      //Lista com a consulta mais recente de compras no servidor.
19      private ArrayList<Compra> compras = new ArrayList<>();
20      private Context context;
21      private static Compra selecionada;
22
23      public ControleDeCompra(Context context){
24          this.context = context;
25      }
26
27      public static void salvarCompra(Compra compra) {
```

```
28          compra.save();
29      }
30
31      public static void excluirCompra(Compra compra) {
32          compra.delete();
33      }
34
35      public void consultarCompra(String consulta, final RequestCallback<Compra> callback) {
36          if (consulta.isEmpty()) {
37              Model.find(context, Compra.class, new TypeToken<ArrayList<Compra>>() {
38              }.getType(), new RequestCallback<Compra>() {
39                  @Override
40                  public void execute(ArrayList<Compra> lista) throws Exception {
41                      super.execute(lista);
42
43                      compras.clear();
44                      compras.addAll(lista);
45
46                      if (callback != null) {
47                          callback.execute(lista);
48                      }
49                  }
50              }, null);
51          } else {
52              try {
53                  ArrayList<Compra> comprasFiltradas = new ArrayList<>();
54
55                  for (Compra compra : compras) {
56                      if (compra.getNomeDoItem().contains(consulta)) {
57                          comprasFiltradas.add(compra);
58                      }
59                  }
60
61                  if (callback != null) {
62                      callback.execute(comprasFiltradas);
63                  }
64              } catch (Exception e) {
65                  Log.e(Utils.TAG, "Erro ao consultar fornecedores: ", e);
66              }
67          }
68      }
69
70      public static void selecionarParaEditar(Compra compra){
71          ControleDeCompra.selecionada = compra;
72      }
73
74      public static Compra getSelecionada(){
75          return selecionada;
76      }
77
78      public static void deselecionar(){
79          selecionada = null;
80      }
81  }
```

Código 15: ControleDeCompra.java

4.2.3 Controle de Fornecedor

```
1   package ds2.equipe1.restaurante.controles;
2
3   import android.content.Context;
4   import android.util.Log;
5   import android.widget.Toast;
6
7   import com.androidquery.callback.AjaxCallback;
8   import com.google.gson.reflect.TypeToken;
9
10  import java.lang.reflect.Array;
11  import java.util.ArrayList;
12
13  import ds2.equipe1.restaurante.helpers.RequestCallback;
14  import ds2.equipe1.restaurante.helpers.Utils;
15  import ds2.equipe1.restaurante.modelos.Fornecedor;
16  import ds2.equipe1.restaurante.modelos.Item;
17  import ds2.equipe1.restaurante.modelos.Model;
18
19  public class ControleDeFornecedor {
20      //Lista com a consulta mais recente de fornecedores no servidor.
21      private ArrayList<Fornecedor> fornecedores = new ArrayList<>();
22      //Essa variavel serve para deixar uma referencia na memoria o tempo inteiro de qual esta selecionado,
23      //para que quando haja uma edicao, a alteracao seja refletida em mais de uma tela (exemplo: editar um fornecedor e
        atualizar na tela de busca).
24      private static Fornecedor selecionado;
25      private Context context;
26
27      public ControleDeFornecedor(Context context){
28          this.context = context;
```

```
    }

    public void salvarFornecedor(Fornecedor fornecedor){
        fornecedor.save();
    }

    public boolean excluirFornecedor(Fornecedor fornecedor){
        if (fornecedor.getId() != -1) {
            fornecedor.delete();
            return true;
        } else return false;
    }

    public void consultarFornecedor(String consulta, final RequestCallback<Fornecedor> callback){
        //Se a consulta for vazio, pega todos os itens do banco de dados, e coloca na memoria ram
        if (consulta.isEmpty()) {
            Model.find(context, Fornecedor.class, new TypeToken<ArrayList<Fornecedor>>() {
                    }.getType(), new RequestCallback<Fornecedor>() {
                        @Override
                        public void execute(ArrayList<Fornecedor> lista) throws Exception {
                            super.execute(lista);

                            fornecedores.clear();
                            fornecedores.addAll(lista);

                            if (callback != null){
                                callback.execute(lista);
                            }
                        }
                    }, null);
        } else {
            //Se tiver consulta, faz a pesquisa nos itens que ja estao na memoria ram
            try {
                ArrayList<Fornecedor> fornecedoresFiltrados = new ArrayList<>();

                for (Fornecedor fornecedor : fornecedores) {
                    if (fornecedor.getNome().contains(consulta) || fornecedor.getCnpj().contains(consulta)) {
                        fornecedoresFiltrados.add(fornecedor);
                    }
                }

                if (callback != null) {
                    callback.execute(fornecedoresFiltrados);
                }
            } catch (Exception e){
                Log.e(Utils.TAG, "Erro ao consultar fornecedores: ", e);
            }
        }
    }

    public void relatorioFornecedor(Fornecedor fornecedor){
        //Banco de dados
        //Exibir Relatorio
    }

    public void relatorioItensPorFornecedor(Item item, Fornecedor fornecedor){
        //Banco de Dados
        //Exibir Relatorio
    }

    public static void selecionarParaEditar(Fornecedor fornecedor){
        ControleDeFornecedor.selecionado = fornecedor;
    }

    public static Fornecedor getSelecionado(){
        return selecionado;
    }

    public static void deselecionar(){
        selecionado = null;
    }
}
```

Código 16: ControleDeFornecedor.java

4.2.4 Controle de Funcionário

```
package ds2.equipe1.restaurante.controles;

import android.content.Context;
import android.util.Log;

import com.google.gson.reflect.TypeToken;

import java.util.ArrayList;

import ds2.equipe1.restaurante.helpers.RequestCallback;
import ds2.equipe1.restaurante.helpers.Utils;
```

```java
import ds2.equipe1.restaurante.modelos.Funcionario;
import ds2.equipe1.restaurante.modelos.Model;

public class ControleDeFuncionario {
    private ArrayList<Funcionario> funcionarios = new ArrayList<>();

    private static Funcionario selecionado;
    private Context context;

    private ControleDeImpressao controleDeImpressao;

    public ControleDeFuncionario(Context context){
        controleDeImpressao = new ControleDeImpressao(context);
    }

    public void salvarFuncionario(Funcionario funcionario){
        funcionario.save();
    }
    public void cadastrarFuncionario(Funcionario funcionario){
        funcionarios.add(funcionario);
        funcionario.save();

        //Banco de Dados
        //SQL (Sem o Endereco)
        //INSERT INTO funcionario (nome, telefone, cnpj, email) VALUES (funcionario.getRua(), funcionario.getTelefone()
 , funcionario.getCnpj(), funcionario.getEmail());
    }

    public boolean alterarFuncionario(Funcionario funcionario){
        if (funcionario.getId() != -1) {
            funcionario.save();
            return true;
        } else return false;

        //Banco de Dados
        //SQL (Sem o Endereco)
        //UPDATE funcionario SET nome = funcionario.getRua(), telefone = funcionario.getRua(), email = funcionario.
 getEmail WHERE cnpj = funcionario.getCnpj();
    }

    public boolean excluirFuncionario(Funcionario funcionario){
        if (funcionario.getId() != -1) {
            funcionario.delete();
            return true;
        } else return false;
    }

    public void consultarFuncionario(String consulta, final RequestCallback<Funcionario> callback){
        //Se a consulta for vazia, pega todos os itens do banco de dados e coloca na memoria ram
        if (consulta.isEmpty()) {
            Model.find(context, Funcionario.class, new TypeToken<ArrayList<Funcionario>>() {
            }.getType(), new RequestCallback<Funcionario>() {
                @Override
                public void execute(ArrayList<Funcionario> lista) throws Exception {
                    super.execute(lista);

                    funcionarios.clear();
                    funcionarios.addAll(lista);

                    if (callback != null){
                        callback.execute(lista);
                    }
                }
            }, null);
        } else {
            //Se tiver consulta, faz a pesquisa nos itens que ja estao na memoria ram
            try {
                ArrayList<Funcionario> funcionariosFiltrados = new ArrayList<>();

                for (Funcionario funcionario : funcionarios) {
                    if (funcionario.getNome().contains(consulta) || funcionario.getCpf().contains(consulta)) {
                        funcionariosFiltrados.add(funcionario);
                    }
                }

                if (callback != null) {
                    callback.execute(funcionariosFiltrados);
                }
            } catch (Exception e){
                Log.e(Utils.TAG, "Erro ao consultar funcionarios: ", e);
            }
        }
    }

    public void relatorioFuncionario(Funcionario funcionario){
        //Banco de dados
        //Exibir Relatorio
    }

    public static void selecionarParaEditar(Funcionario funcionario){
        ControleDeFuncionario.selecionado = funcionario;
    }

    public static Funcionario getSelecionado(){
```

```
104         return selecionado;
105     }
106
107     public static void deselecionar(){
108         selecionado = null;
109     }
110
111 }
```

Código 17: ControleDeFuncionario.java

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  public class ControleDeImpressao {
9      private Context context;
10
11     public ControleDeImpressao(Context context) {
12         this.context = context;
13     }
14
15     public void imprimir(String dados){
16         Intent i = new Intent(context, TelaRelatorios.class);
17         i.putExtra("dados", dados);
18         context.startActivity(i);
19     }
20 }
```

Código 18: ControleDeImpressao.java

4.2.6 Controle de Item

```
1  package ds2.equipe1.restaurante.controles;
2
3  import android.content.Context;
4  import android.util.Log;
5
6  import com.google.gson.reflect.TypeToken;
7
8  import java.util.ArrayList;
9
10 import ds2.equipe1.restaurante.helpers.RequestCallback;
11 import ds2.equipe1.restaurante.helpers.Utils;
12 import ds2.equipe1.restaurante.modelos.Fornecedor;
13 import ds2.equipe1.restaurante.modelos.Item;
14 import ds2.equipe1.restaurante.modelos.Model;
15
16 public class ControleDeItem {
17     //Lista com a consulta mais recente de itens no servidor.
18     private ArrayList<Item> itens = new ArrayList<>();
19     private Context context;
20     private static Item selecionado;
21
22     public ControleDeItem(Context context) {
23         this.context = context;
24     }
25
26     public static void salvarItem(Item item) {
27         item.save();
28     }
29
30     public static void alterarItemQuantidade(Item item, int quantidade) {
31         item.setQuantidade(quantidade);
32         salvarItem(item);
33     }
34
35     public static void alterarItemLimiteMinimo(Item item, int limiteMinimo) {
36         item.setLimiteMinimo(limiteMinimo);
37         salvarItem(item);
38     }
39
40     public static void alterarItem(Item item, int quantidade, int limiteMinimo) {
41         if (quantidade >= 0) {
42             item.setQuantidade(quantidade);
43         }
44
```

```
45          if (limiteMinimo >= 0) {
46              item.setLimiteMinimo(limiteMinimo);
47          }
48
49          salvarItem(item);
50      }
51
52      public static void excluirItem(Item item) {
53          item.delete();
54      }
55
56      public void consultarItem(String consulta, final RequestCallback<Item> callback) {
57          //Se a consulta for vazio, pega todos os itens do banco de dados, e coloca na memoria ram
58          if (consulta.isEmpty()) {
59              Model.find(context, Item.class, new TypeToken<ArrayList<Item>>() {
60              }.getType(), new RequestCallback<Item>() {
61                  @Override
62                  public void execute(ArrayList<Item> lista) throws Exception {
63                      super.execute(lista);
64
65                      itens.clear();
66                      itens.addAll(lista);
67
68                      if (callback != null) {
69                          callback.execute(lista);
70                      }
71                  }
72              }, null);
73          } else {
74              //Se tiver consulta, faz a pesquisa nos itens que ja estao na memoria ram
75              try {
76                  ArrayList<Item> itensFiltrados = new ArrayList<>();
77
78                  for (Item item : itens) {
79                      if (item.getNome().contains(consulta)) {
80                          itensFiltrados.add(item);
81                      }
82                  }
83
84                  if (callback != null) {
85                      callback.execute(itensFiltrados);
86                  }
87              } catch (Exception e) {
88                  Log.e(Utils.TAG, "Erro ao consultar fornecedores: ", e);
89              }
90          }
91      }
92
93      public static void relatorioItens(ArrayList<Item> itens) {
94          for (int i = 0; i < itens.size(); i++) {
95              Item item = itens.get(i);
96              // TODO: alterar para exibicao na tela do android ou gerar pdf
97              System.out.println(
98                      "Item: " + item.getNome() +
99                              ", Quantidade Disponivel: " + item.getQuantidade() +
100                             ", Limite Minimo: " + item.getLimiteMinimo()
101             );
102         }
103     }
104
105     public static void relatorioItensEmFalta(ArrayList<Item> itens) {
106         for (int i = 0; i < itens.size(); i++) {
107             Item item = itens.get(i);
108             if (item.getQuantidade() < item.getLimiteMinimo()) {
109                 // TODO: alterar para exibicao na tela do android ou gerar pdf
110                 System.out.println(
111                         "Item em Falta: " + item.getNome() +
112                                 ", Quantidade Disponivel: " + item.getQuantidade() +
113                                 ", Limite Minimo: " + item.getLimiteMinimo()
114                 );
115             }
116         }
117     }
118
119     public static void selecionarParaEditar(Item item){
120         ControleDeItem.selecionado = item;
121     }
122
123     public static Item getSelecionado(){
124         return selecionado;
125     }
126
127     public static void deselecionar(){
128         selecionado = null;
129     }
130
131 }
```

Código 19: ControleDeItem.java

4.2.7 Controle de Produto

```java
package ds2.equipe1.restaurante.controles;

import android.content.Context;

import java.util.ArrayList;

import ds2.equipe1.restaurante.modelos.Ingrediente;
import ds2.equipe1.restaurante.modelos.Item;
import ds2.equipe1.restaurante.modelos.Produto;
import java.sql.*;

public class ControleDeProduto {

    private ArrayList <Produto> produtos;
    private Context context;

    public ControleDeProduto(Context context){
        this.context = context;
    }


    public void carregarProdutosDoBanco(){
        produtos = new ArrayList <Produto>();
        int tamanhoDoCardapio = 5; //Ler o tamanho do banco

        for (int i = 0; i < tamanhoDoCardapio; i++) {
            String nome = "teste"; //Ler o nome do banco
            //produtos.add(new Produto(nome));
        }
    }

    public void cadastrarProduto(String nome, float preco, ArrayList <Ingrediente> ingredientes){
        produtos.add(new Produto(context, nome, preco, ingredientes));
    }

    public void excluirProduto(String nome){
        for (int i = 0; i < produtos.size(); i++) {
            if (produtos.get(i).getNome() == nome) {
                //Remover do banco
                produtos.remove(i); //ver se e este mesmo o metodo de remocao
            }
        }
    }

    public void alterarPrecoDeProduto(String nome, float novoPreco){
        for (int i = 0; i < produtos.size(); i++) {
            if (produtos.get(i).getNome() == nome) {
                //Alterar preco no banco
                produtos.get(i).setPreco(novoPreco);
            }
        }
    }

    public Produto consultarProduto(String nome){
        for (int i = 0; i < produtos.size(); i++) {
            if (produtos.get(i).getNome() == nome) {
                return produtos.get(i);
            }
        }
        return null;
    }

    public void relatorioProdutos(){
        //Gerar pdf?
    }

    public void relatorioItensPorProduto(){
        //Gerar pdf?
    }

}
```

Código 20: ControleDeProduto.java

4.3 Fronteiras

4.3.1 Main Activity

```java
package ds2.equipe1.restaurante;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

```

```
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 21: 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              utils.inputDialog("Configurar host", utils.getData("host", ""), "http://192.168.1.100/restaurante/", new
     Utils.DialogCallback() {
83                  @Override
84                  public void execute(String text) {
85                      utils.setData("host", text);
86                  }
87              });
88              return true;
89          }
90
91          return super.onOptionsItemSelected(item);
92      }
93
94      public void open(Class activity){
95          this.startActivity(new Intent(this, activity));
96      }
97  }
```

Código 22: 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
16          findViewById(R.id.menu_cadastrar_fornecedor).setOnClickListener(new View.OnClickListener() {
17              @Override
18              public void onClick(View v) {
19                  open(CadastroFornecedor.class);
20              }
21          });
22
23          findViewById(R.id.menu_buscar_fornecedor).setOnClickListener(new View.OnClickListener() {
24              @Override
25              public void onClick(View v) {
26                  open(BuscaFornecedor.class);
27              }
28          });
29      }
30
31      public void open(Class activity){
32          startActivity(new Intent(this, activity));
33      }
34  }
```

Código 23: 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          });
35      }
36
37      public void open(Class activity){
38          startActivity(new Intent(this, activity));
39      }
40  }
```

Código 24: 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",
        Toast.LENGTH_LONG).show();
34              }
35          });
36      }
37
38      public void open(Class activity){
39          startActivity(new Intent(this, activity));
40      }
41  }
```

Código 25: 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 MenuItens 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 26: 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 27: MenuProdutos.java

4.3.8 Menu Relatório

```
package ds2.equipe1.restaurante;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

public class MenuRelatorio extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_menu_relatorio);
    }
}
```

Código 28: MenuRelatorio.java

4.3.9 Busca Fornecedor

```
package ds2.equipe1.restaurante;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.Toast;

import java.util.ArrayList;

import ds2.equipe1.restaurante.controles.ControleDeFornecedor;
import ds2.equipe1.restaurante.helpers.RequestCallback;
import ds2.equipe1.restaurante.helpers.Utils;
import ds2.equipe1.restaurante.listas.FornecedorAdapter;
import ds2.equipe1.restaurante.modelos.Fornecedor;

public class BuscaFornecedor extends AppCompatActivity {
    private ControleDeFornecedor controleDeFornecedor;

    private ListView lvFornecedores;
    private FornecedorAdapter adapter;
    //Fornecedores visiveis na tela
    private ArrayList<Fornecedor> fornecedores;
    private EditText edtProcurar;
    private ImageView ivProcurar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_busca_fornecedor);

        init();

        controleDeFornecedor = new ControleDeFornecedor(this);

        fornecedores = new ArrayList<>();
        adapter = new FornecedorAdapter(BuscaFornecedor.this, fornecedores);
        lvFornecedores.setAdapter(adapter);

        consultar("");
    }

    private void init(){
        //pegar referencias dos componentes xml
        lvFornecedores = (ListView) findViewById(R.id.lvFornecedores);
        edtProcurar = (EditText) findViewById(R.id.edtProcurar);
        ivProcurar = (ImageView) findViewById(R.id.ivProcurar);

        //atribuir funcionalidades para alguns componentes
        ivProcurar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                String texto = edtProcurar.getText().toString();

                consultar(texto);
            }
        });

        lvFornecedores.setOnItemClickListener(new AdapterView.OnItemClickListener() {
            @Override
            // no clique do item o android nos da a view, a posicao do item na lista e o ID do item
            // (que nos configuramos) entao passamos esse ID para
            public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
```

```
68                  ControleDeFornecedor.selecionarParaEditar(fornecedores.get(position));
69
70                  Intent intent = new Intent(BuscaFornecedor.this, CadastroFornecedor.class);
71                  intent.putExtra("alterar", true);
72                  startActivity(intent);
73              }
74          });
75      }
76
77      private void consultar(final String consulta){
78          controleDeFornecedor.consultarFornecedor(consulta, new RequestCallback<Fornecedor>(this) {
79              @Override
80              public void execute(ArrayList<Fornecedor> fornecedores) throws Exception {
81                  super.execute(fornecedores);
82
83                  BuscaFornecedor.this.fornecedores.clear();
84                  BuscaFornecedor.this.fornecedores.addAll(fornecedores);
85
86                  adapter.notifyDataSetChanged();
87              }
88          });
89      }
90
91      @Override
92      protected void onResume() {
93          //verificar se o fornecedor foi excluido para remover da tela.
94          for (int i = 0; i < fornecedores.size(); i++){
95              if (fornecedores.get(i).getId() == null){
96                  fornecedores.remove(i--);
97              }
98          }
99
100         //Quando a tela reabrir, atualizar a interface com as informacoes alteradas do item selecionado.
101         adapter.notifyDataSetChanged();
102
103         super.onResume();
104     }
105 }
```

Código 29: BuscaFornecedor.java

4.3.10 Busca Funcionário

```
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.AdapterView;
8  import android.widget.EditText;
9  import android.widget.ImageView;
10 import android.widget.ListView;
11
12 import java.util.ArrayList;
13
14 import ds2.equipe1.restaurante.controles.ControleDeFuncionario;
15 import ds2.equipe1.restaurante.helpers.RequestCallback;
16 import ds2.equipe1.restaurante.listas.FuncionarioAdapter;
17 import ds2.equipe1.restaurante.modelos.Funcionario;
18
19 public class BuscaFuncionario extends AppCompatActivity {
20     public static ControleDeFuncionario controleDeFuncionario;
21
22     private ListView lvFuncionarios;
23     private FuncionarioAdapter adapter;
24     private ArrayList<Funcionario> funcionarios;
25     private EditText edtProcurar;
26     private ImageView ivProcurar;
27
28     @Override
29     protected void onCreate(Bundle savedInstanceState) {
30         super.onCreate(savedInstanceState);
31         setContentView(R.layout.activity_busca_funcionario);
32
33         init();
34
35         controleDeFuncionario = new ControleDeFuncionario(this);
36
37         funcionarios = new ArrayList<>();
38         adapter = new FuncionarioAdapter(BuscaFuncionario.this, funcionarios);
39         lvFuncionarios.setAdapter(adapter);
40
41         consultar("");
42     }
43     private void init(){
44         //pegar referencias dos componentes xml
45         lvFuncionarios = (ListView) findViewById(R.id.lvFuncionarios);
```

```
46          edtProcurar = (EditText) findViewById(R.id.edtProcurar);
47          ivProcurar = (ImageView) findViewById(R.id.ivProcurar);
48
49          //atribuir funcionalidades para alguns componentes
50          ivProcurar.setOnClickListener(new View.OnClickListener() {
51              @Override
52              public void onClick(View v) {
53                  String texto = edtProcurar.getText().toString();
54
55                  consultar(texto);
56              }
57          });
58
59          lvFuncionarios.setOnItemClickListener(new AdapterView.OnItemClickListener() {
60              @Override
61              //no clique do item o android nos da a view, a posicao do item na lista e o ID do item (que nos
        configuramos)
62              //entao passamos esse ID para
63              public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
64                  ControleDeFuncionario.selecionarParaEditar(funcionarios.get(position));
65
66                  Intent intent = new Intent(BuscaFuncionario.this, CadastroFuncionario.class);
67                  intent.putExtra("alterar", true);
68                  startActivity(intent);
69              }
70          });
71      }
72
73      private void consultar(final String consulta){
74          controleDeFuncionario.consultarFuncionario(consulta, new RequestCallback<Funcionario>(this) {
75              @Override
76              public void execute(ArrayList<Funcionario> funcionarios) throws Exception {
77                  super.execute(funcionarios);
78
79                  BuscaFuncionario.this.funcionarios.clear();
80                  BuscaFuncionario.this.funcionarios.addAll(funcionarios);
81
82                  adapter.notifyDataSetChanged();
83              }
84          });
85      }
86
87      @Override
88      protected void onResume() {
89          //verificar se o Funcionario foi excluido para remover da tela.
90          for (int i = 0; i < funcionarios.size(); i++){
91              if (funcionarios.get(i).getId() == null){
92                  funcionarios.remove(i--);
93              }
94          }
95
96          //Quando a tela reabrir, atualizar a interface com as informacoes alteradas do item selecionado.
97          adapter.notifyDataSetChanged();
98
99          super.onResume();
100     }
101
102
103
104     public void onItemClick(View v){
105         startActivity(new Intent(this, CadastroFuncionario.class));
106     }
107
108
109 }
```

Código 30: BuscaFuncionario.java

4.3.11 Busca Item

```
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.AdapterView;
8  import android.widget.EditText;
9  import android.widget.ImageView;
10 import android.widget.ListView;
11 import android.widget.Toast;
12
13 import java.util.ArrayList;
14
15 import ds2.equipe1.restaurante.controles.ControleDeItem;
16 import ds2.equipe1.restaurante.helpers.RequestCallback;
17 import ds2.equipe1.restaurante.listas.ItemAdapter;
18 import ds2.equipe1.restaurante.modelos.Item;
```

```
19
20
21  public class BuscaItem extends AppCompatActivity {
22
23      private ControleDeItem controleDeItem;
24
25      private ListView lvItens;
26      private ItemAdapter adapter;
27
28      private ArrayList<Item> itens;
29      private EditText edtProcurar;
30      private ImageView ivProcurar;
31
32      @Override
33      protected void onCreate(Bundle savedInstanceState) {
34          super.onCreate(savedInstanceState);
35          setContentView(R.layout.activity_busca_fornecedor);
36
37          init();
38
39          controleDeItem = new ControleDeItem(this);
40
41          itens = new ArrayList<>();
42          adapter = new ItemAdapter(BuscaItem.this, itens);
43          lvItens.setAdapter(adapter);
44
45          consultar("");
46      }
47
48      private void init(){
49          //pegar referencias dos componentes xml
50          lvItens = (ListView) findViewById(R.id.lvFornecedores);
51          edtProcurar = (EditText) findViewById(R.id.edtProcurar);
52          ivProcurar = (ImageView) findViewById(R.id.ivProcurar);
53
54          //atribuir funcionalidades para alguns componentes
55          ivProcurar.setOnClickListener(new View.OnClickListener() {
56              @Override
57              public void onClick(View v) {
58                  String texto = edtProcurar.getText().toString();
59
60                  consultar(texto);
61
62                  if (itens.size() == 0){
63                      Toast.makeText(BuscaItem.this, "Nenhum␣resultado␣para␣\"" + texto + "\"", Toast.LENGTH_SHORT).show
        ();
64                  }
65              }
66          });
67
68          lvItens.setOnItemClickListener(new AdapterView.OnItemClickListener() {
69              @Override
70              public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
71                  Intent intent = new Intent(BuscaItem.this, CadastroItem.class);
72                  Toast.makeText(BuscaItem.this, "" + id, Toast.LENGTH_SHORT).show();
73                  intent.putExtra("id", id);
74                  startActivity(intent);
75              }
76          });
77      }
78
79      private void consultar(String consulta){
80          controleDeItem.consultarItem(consulta, new RequestCallback<Item>(){
81              @Override
82              public void execute(ArrayList<Item> itens) throws Exception {
83                  BuscaItem.this.itens.clear();
84                  BuscaItem.this.itens.addAll(itens);
85                  adapter.notifyDataSetChanged();
86
87                  super.execute(itens);
88              }
89          });
90      }
91  }
```

Código 31: BuscaItem.java

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
```

```
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_busca_produto);
    }

    public void onItemClick(View v){
        startActivity(new Intent(this, CadastroProduto.class));
    }
}
```

Código 32: BuscaProduto.java

4.3.13 Cadastro Endereço

```
package ds2.equipe1.restaurante;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import org.json.JSONObject;

import ds2.equipe1.restaurante.helpers.RequestCallback;
import ds2.equipe1.restaurante.modelos.Endereco;
import ds2.equipe1.restaurante.modelos.Model;

public class CadastroEndereco extends AppCompatActivity {

    private EditText edtCEP, edtLogradouro, edtRua, edtBairro, edtCidade, edtEstado, edtNumero;
    private Button btnCadastrar;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cadastro_endereco);

        init();
    }

    private void init(){
        btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
        edtRua = (EditText) findViewById(R.id.edtRua);
        edtCEP = (EditText) findViewById(R.id.edtCEP);
        edtLogradouro = (EditText) findViewById(R.id.edtLogradouro);
        edtBairro = (EditText) findViewById(R.id.edtBairro);
        edtCidade = (EditText) findViewById(R.id.edtCidade);
        edtEstado = (EditText) findViewById(R.id.edtEstado);
        edtNumero = (EditText) findViewById(R.id.edtNumero);

        btnCadastrar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {

                final Endereco endereco = new Endereco(CadastroEndereco.this, edtLogradouro.getText().toString(),
     edtRua.getText().toString(), Integer.parseInt(edtNumero.getText().toString()), edtBairro.getText().toString(),
     edtCidade.getText().toString(), edtEstado.getText().toString(), edtCEP.getText().toString());

                endereco.save(new RequestCallback<Model>() {
                    @Override
                    public void execute(Model model) throws Exception {
                        super.execute(model);

                        Intent i = getIntent();
                        i.putExtra("rua", endereco.getRua());
                        i.putExtra("id_endereco", model.getId());
                        setResult(RESULT_OK, i);
                        finish();
                    }
                });
            }
        });
    }
}
```

Código 33: CadastroEndereco.java

4.3.14 Cadastro Fornecedor

```java
package ds2.equipe1.restaurante;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import ds2.equipe1.restaurante.controles.ControleDeFornecedor;
import ds2.equipe1.restaurante.helpers.Utils;
import ds2.equipe1.restaurante.modelos.Fornecedor;

public class CadastroFornecedor extends AppCompatActivity {
    private ControleDeFornecedor controleDeFornecedor;
    private EditText edtNome, edtCNPJ, edtEndereco, edtEmail, edtTelefone;
    private Button btnCadastrar, btnCadastrarEndereco, btnExcluir;

    private Fornecedor fornecedor;
    private boolean novoCadastro = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cadastro_fornecedor);

        init();

        fornecedor = new Fornecedor(this);
        controleDeFornecedor = new ControleDeFornecedor(this);

        Intent intent = getIntent();
        if (intent.getBooleanExtra("alterar", false)){
            novoCadastro = false;
            carregarFornecedor();
        }

        if (novoCadastro){
            btnExcluir.setVisibility(View.GONE);
        } else {
            btnCadastrar.setText("Alterar");
            btnExcluir.setVisibility(View.VISIBLE);
        }
    }

    private void init(){
        edtNome = (EditText) findViewById(R.id.edtNome);
        edtCNPJ = (EditText) findViewById(R.id.edtCNPJ);
        edtEndereco = (EditText) findViewById(R.id.edtEndereco);
        edtEmail = (EditText) findViewById(R.id.edtEmail);
        edtTelefone = (EditText) findViewById(R.id.edtTelefone);
        btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
        btnCadastrarEndereco = (Button) findViewById(R.id.btnCadastrarEndereco);
        btnExcluir = (Button) findViewById(R.id.btnExcluir);

        btnCadastrarEndereco.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onCadastrarEnderecoClick();
            }
        });

        btnExcluir.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if (!novoCadastro && fornecedor.getId() != null) {
                    fornecedor.delete();
                    fornecedor.setId(null);
                    new Utils(CadastroFornecedor.this).toast("Fornecedor excluido!");
                    finish();
                }
            }
        });

        btnCadastrar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onCadastrarClick();
            }
        });
    }

    private void onCadastrarEnderecoClick(){
        Intent intent = new Intent(this, CadastroEndereco.class);
        startActivityForResult(intent,1);
    }

    private void onCadastrarClick(){
        final String nome = edtNome.getText().toString();
        final String CNPJ = edtCNPJ.getText().toString();
        final String email = edtEmail.getText().toString();
        final String telefone = edtTelefone.getText().toString();

```

```
/*        endereco.save(new RequestCallback() {
            @Override
            public void execute() {
                controleDeFornecedor.salvarFornecedor(fornecedor);
            }
        });*/

        fornecedor.setNome(nome);
        fornecedor.setCnpj(CNPJ);
        fornecedor.setEmail(email);
        fornecedor.setTelefone(telefone);
        controleDeFornecedor.salvarFornecedor(fornecedor);

        if (fornecedor.getId() == null) {
            new Utils(this).toast("Fornecedor cadastrado!");
        } else {
            new Utils(this).toast("Fornecedor alterado!");
        }

        ControleDeFornecedor.deselecionar();
        finish();
    }

    public void carregarFornecedor(){
        if (ControleDeFornecedor.getSelecionado() != null) {
            CadastroFornecedor.this.fornecedor = ControleDeFornecedor.getSelecionado();

            edtNome.setText(fornecedor.getNome());
            edtCNPJ.setText(fornecedor.getCnpj());
            edtTelefone.setText(fornecedor.getTelefone());
            edtEmail.setText(fornecedor.getEmail());
        }
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == 1 && resultCode == RESULT_OK){
            if (data.hasExtra("rua")) {
                edtEndereco.setText(data.getStringExtra("rua"));
            }
            /*if (data.hasExtra("id_endereco")) {
                fornecedor.setIdEndereco(data.getIntExtra("id_endereco", -1));
            }*/
        }
    }
}
```

Código 34: CadastroFornecedor.java

4.3.15 Cadastro Funcionário

```
package ds2.equipe1.restaurante;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;

import ds2.equipe1.restaurante.controles.ControleDeFuncionario;
import ds2.equipe1.restaurante.helpers.Utils;
import ds2.equipe1.restaurante.modelos.Funcionario;

public class CadastroFuncionario extends AppCompatActivity {

    private ControleDeFuncionario controleDeFuncionario;
    private EditText edtNome, edtCPF, edtEndereco, edtTelefone, edtNome_de_usuario;
    private Button btnCadastrar, btnCadastrarEndereco, btnExcluir;

    private Funcionario funcionario;
    private boolean novoCadastro = true;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_cadastro_funcionario);

        init();

        funcionario = new Funcionario(this);
        controleDeFuncionario = new ControleDeFuncionario(this);

        Intent intent = getIntent();
        if (intent.getBooleanExtra("alterar", false)){
            novoCadastro = false;
```

```
36              carregarFuncionario();
37          }
38
39          if (novoCadastro){
40              btnExcluir.setVisibility(View.GONE);
41          } else {
42              btnCadastrar.setText("Alterar");
43              btnExcluir.setVisibility(View.VISIBLE);
44          }
45      }
46
47      private void init(){
48          edtNome = (EditText) findViewById(R.id.edtNome);
49          edtCPF = (EditText) findViewById(R.id.edtCPF);
50          edtNome_de_usuario = (EditText) findViewById(R.id.edtNome_de_usuario);
51          edtEndereco = (EditText) findViewById(R.id.edtEndereco);
52          edtTelefone = (EditText) findViewById(R.id.edtTelefone);
53          btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
54          btnCadastrarEndereco = (Button) findViewById(R.id.btnCadastrarEndereco);
55          btnExcluir = (Button) findViewById(R.id.btnExcluir);
56
57          btnCadastrarEndereco.setOnClickListener(new View.OnClickListener() {
58              @Override
59              public void onClick(View v) {
60                  onCadastrarEnderecoClick();
61              }
62          });
63
64          btnExcluir.setOnClickListener(new View.OnClickListener() {
65              @Override
66              public void onClick(View v) {
67                  if (!novoCadastro && funcionario.getId() != null) {
68                      funcionario.delete();
69                      funcionario.setId(null);
70                      new Utils(CadastroFuncionario.this).toast("Funcionario␣excluido!");
71                      finish();
72                  }
73              }
74          });
75
76          btnCadastrar.setOnClickListener(new View.OnClickListener() {
77              @Override
78              public void onClick(View v) {
79                  onCadastrarClick();
80              }
81          });
82      }
83
84      private void onCadastrarEnderecoClick(){
85          Intent intent = new Intent(this, CadastroEndereco.class);
86          startActivityForResult(intent,1);
87      }
88
89      private void onCadastrarClick(){
90          final String nome = edtNome.getText().toString();
91          final String CPF = edtCPF.getText().toString();
92          final String telefone = edtTelefone.getText().toString();
93          final String nome_de_usuario = edtNome_de_usuario.getText().toString();
94
95          funcionario.setNome(nome);
96          funcionario.setCpf(CPF);
97          funcionario.setNome_de_usuario(nome_de_usuario);
98          funcionario.setTelefone(telefone);
99          controleDeFuncionario.salvarFuncionario(funcionario);
100
101         if (funcionario.getId() == null) {
102             new Utils(this).toast("Funcionario␣cadastrado!");
103         } else {
104             new Utils(this).toast("Funcionario␣alterado!");
105         }
106
107         ControleDeFuncionario.deselecionar();
108         finish();
109     }
110
111     public void carregarFuncionario(){
112         if (ControleDeFuncionario.getSelecionado() != null) {
113             CadastroFuncionario.this.funcionario = ControleDeFuncionario.getSelecionado();
114
115             edtNome.setText(funcionario.getNome());
116             edtCPF.setText(funcionario.getCpf());
117             edtNome_de_usuario.setText(funcionario.getNome_de_usuario());
118             edtTelefone.setText(funcionario.getTelefone());
119         }
120     }
121
122     @Override
123     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
124         super.onActivityResult(requestCode, resultCode, data);
125
126         if (requestCode == 1 && resultCode == RESULT_OK){
127             if (data.hasExtra("rua")) {
128                 edtEndereco.setText(data.getStringExtra("rua"));
129             }
```

```
130             if (data.hasExtra("id_endereco")) {
131                 funcionario.setIdEndereco(data.getIntExtra("id_endereco", -1));
132             }
133         }
134     }
135 }
```

Código 35: CadastroFuncionario.java

4.3.16 Cadastro Item

```
1  package ds2.equipe1.restaurante;
2
3  import android.content.Intent;
4  import android.support.annotation.StringDef;
5  import android.support.v7.app.AppCompatActivity;
6  import android.os.Bundle;
7  import android.view.View;
8  import android.widget.Button;
9  import android.widget.EditText;
10
11 import ds2.equipe1.restaurante.controles.ControleDeItem;
12 import ds2.equipe1.restaurante.helpers.Utils;
13 import ds2.equipe1.restaurante.modelos.Item;
14
15 public class CadastroItem extends AppCompatActivity {
16
17     private EditText edtNome,edtQuantidade,edtUnidade,edtLimiteMinimo;
18     private Button btnCadastrar, btnExcluir;
19     private ControleDeItem controleDeItem;
20     private Item item;
21
22     private boolean novoCadastro = true;
23
24     @Override
25     protected void onCreate(Bundle savedInstanceState) {
26         super.onCreate(savedInstanceState);
27         setContentView(R.layout.activity_cadastro_item);
28
29         init();
30
31         item = new Item(this);
32         controleDeItem = new ControleDeItem(this);
33
34         Intent intent = getIntent();
35         if (intent.getBooleanExtra("alterar", false)){
36             novoCadastro = false;
37             carregarItem();
38         }
39
40         if (novoCadastro){
41             btnExcluir.setVisibility(View.GONE);
42         } else {
43             btnCadastrar.setText("Alterar");
44             btnExcluir.setVisibility(View.VISIBLE);
45         }
46     }
47
48     private void init(){
49         edtNome = (EditText) findViewById(R.id.edtNome);
50         edtQuantidade = (EditText) findViewById(R.id.edtQuantidade);
51         edtUnidade = (EditText) findViewById(R.id.edtUnidade);
52         edtLimiteMinimo = (EditText) findViewById(R.id.edtLimiteMinimo);
53         btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
54         btnExcluir = (Button) findViewById(R.id.btnExcluir);
55
56         btnExcluir.setOnClickListener(new View.OnClickListener() {
57             @Override
58             public void onClick(View v) {
59                 if (!novoCadastro && item.getId() != null) {
60                     item.delete();
61                     item.setId(null);
62                     new Utils(CadastroItem.this).toast("Item excluido!");
63                     finish();
64                 }
65             }
66         });
67
68         btnCadastrar.setOnClickListener(new View.OnClickListener() {
69             @Override
70             public void onClick(View v) {
71                 onCadastrarClick();
72             }
73         });
74     }
75
76
77     private void onCadastrarClick() {
```

```
78          final String nome = edtNome.getText().toString();
79          final int quantidade = Integer.parseInt(edtQuantidade.getText().toString());
80          final String unidade = edtUnidade.getText().toString();
81          final int limite = Integer.parseInt(edtLimiteMinimo.getText().toString());
82
83          item.setNome(nome);
84          item.setQuantidade(quantidade);
85          item.setUnidade(unidade);
86          item.setLimiteMinimo(limite);
87
88          controleDeItem.salvarItem(item);
89
90          if (item.getId() == null) {
91              new Utils(this).toast("Item cadastrado!");
92          } else {
93              new Utils(this).toast("Item alterado!");
94          }
95
96      }
97
98      private void carregarItem(){
99          if (ControleDeItem.getSelecionado() != null) {
100             CadastroItem.this.item = ControleDeItem.getSelecionado();
101
102             edtNome.setText(item.getNome());
103             edtQuantidade.setText(item.getQuantidade());
104             edtUnidade.setText(item.getQuantidade());
105             edtLimiteMinimo.setText(item.getLimiteMinimo());
106         }
107     }
108
109
110 }
```

Código 36: CadastroItem.java

4.3.17 Cadastro Produto

```
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.View;
8  import android.widget.Button;
9  import android.widget.EditText;
10 import android.widget.ImageButton;
11
12 import java.util.ArrayList;
13
14 import ds2.equipe1.restaurante.controles.ControleDeProduto;
15 import ds2.equipe1.restaurante.helpers.Utils;
16 import ds2.equipe1.restaurante.modelos.Ingrediente;
17 import ds2.equipe1.restaurante.modelos.Produto;
18
19 public class CadastroProduto extends AppCompatActivity {
20
21     private ControleDeProduto controleDeProduto;
22     private EditText edtNome, edtPreco;
23     private Button btnCadastrar, btnCancelar;
24     private ImageButton btnAddIngrediente;
25     private ArrayList<Ingrediente> ingredientes = new ArrayList<Ingrediente>();
26     private Context context;
27
28     @Override
29     protected void onCreate(Bundle savedInstanceState) {
30         super.onCreate(savedInstanceState);
31         setContentView(R.layout.activity_cadastro_produto);
32         //this.context = ??? PEGAR CONTEXTO;
33
34         init();
35
36         controleDeProduto = new ControleDeProduto(this);
37     }
38
39     private void init(){
40         edtNome = (EditText) findViewById(R.id.edtNome);
41         edtPreco = (EditText) findViewById(R.id.edtPreco);
42         btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
43         btnCancelar = (Button) findViewById(R.id.btnCancelar);
44         btnAddIngrediente = (ImageButton) findViewById(R.id.btnAddIngrediente);
45
46         btnCadastrar.setOnClickListener(new View.OnClickListener() {
47             @Override
48             public void onClick(View v) {
49                 onCadastrarClick();
50             }
```

```
        });

        btnCancelar.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onCancelarClick();
            }
        });

        btnAddIngrediente.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onAddIngredienteClick();
            }
        });
    }

    private void onCadastrarClick(){
        final String nome = edtNome.getText().toString();
        final Float preco = Float.parseFloat(edtPreco.getText().toString());

        Produto produto = new Produto(context,nome,preco,ingredientes);


        new Utils(this).toast("Produto cadastrado!");
        finish();
    }

    private void onCancelarClick(){
        finish();
    }

    private void onAddIngredienteClick(){
        new Utils(this).selectPopup("Cadastrar ingrediente", null);

        //Intent intent = new Intent(this, SelecionarIngredientes.class);
        //startActivityForResult(intent,1);

        //Implementar lista de selecao de ingredientes
    }
}
```

Código 37: CadastroProduto.java

4.3.18 Criação Comanda

```
package ds2.equipe1.restaurante;

import android.content.Context;
import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

import ds2.equipe1.restaurante.controles.ControleDeAtendimento;
import ds2.equipe1.restaurante.modelos.Comanda;

public class CriacaoComanda extends AppCompatActivity {

    private ControleDeAtendimento controleDeAtendimento;
    Comanda comanda;
    private Context context;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_criacao_comanda);
        controleDeAtendimento = new ControleDeAtendimento(this);

        findViewById(R.id.btnAdicionarPedido).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                startActivity(new Intent(CriacaoComanda.this, BuscaProduto.class));
            }
        });

        findViewById(R.id.btnEncerrarComanda).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                //encerrar comanda
                //imprimir comanda
            }
        });
    }

```

```
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 38: 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 39: ExibirPedido.java

4.3.20 Selecionar Ingredientes

```
1   package ds2.equipe1.restaurante;
2
3   import android.support.v7.app.AppCompatActivity;
4
5   public class SelecionarIngredientes extends AppCompatActivity {
6
7   }
```

Código 40: SelecionarIngredientes.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
```

```
            public void onClick(View v) {
                open(CriacaoComanda.class);
            }
        });

        findViewById(R.id.menu_consulta_cardapio).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                open(BuscaProduto.class);
            }
        });
    }

    public void open(Class activity){
        startActivity(new Intent(this, activity));
    }
}
```

Código 41: TelaGarcom.java

4.3.22 Tela Relatórios

```
package ds2.equipe1.restaurante;

import android.content.Intent;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.widget.TextView;

public class TelaRelatorios extends AppCompatActivity {
    private TextView tvImpressora;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_tela_relatorios);
        init();

        Intent i = getIntent();

        if (i.hasExtra("dados")){
            tvImpressora.setText(i.getStringExtra("dados"));
        } else {
            tvImpressora.setText("Nenhum␣dado␣passado␣como␣parametro");
        }
    }

    private void init(){
        tvImpressora = (TextView) findViewById(R.id.tvImpressora);
    }
}
```

Código 42: TelaRelatorios.java

4.4 Adapters

4.4.1 Fornecedor Adapter

```
package ds2.equipe1.restaurante.listas;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.EditText;
import android.widget.TextView;

import java.util.ArrayList;

import ds2.equipe1.restaurante.R;
import ds2.equipe1.restaurante.modelos.Fornecedor;

public class FornecedorAdapter extends BaseAdapter {
    private Context context;
    private ArrayList<Fornecedor> fornecedores;

    public FornecedorAdapter(Context context, ArrayList<Fornecedor> fornecedores) {
        this.context = context;
        this.fornecedores = fornecedores;
    }
```

```
24
25      @Override
26      public int getCount() {
27          return fornecedores.size();
28      }
29
30      @Override
31      public Fornecedor getItem(int position) {
32          return fornecedores.get(position);
33      }
34
35      @Override
36      public long getItemId(int position) {
37          return fornecedores.get(position).getId();
38      }
39
40      @Override
41      public View getView(int position, View root, ViewGroup parent) {
42          if (root == null) {
43              LayoutInflater vi = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
44              root = vi.inflate(R.layout.item_fornecedor, parent, false);
45          }
46
47          TextView edtNome, edtCNPJ, edtTelefone;
48          edtNome = (TextView) root.findViewById(R.id.tvNome);
49          edtCNPJ = (TextView) root.findViewById(R.id.tvCNPJ);
50          edtTelefone = (TextView) root.findViewById(R.id.tvTelefone);
51
52          Fornecedor fornecedor = fornecedores.get(position);
53
54          edtNome.setText(fornecedor.getNome());
55          edtCNPJ.setText(fornecedor.getCnpj());
56          edtTelefone.setText(fornecedor.getTelefone());
57
58          return root;
59      }
60  }
```

Código 43: 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  public class FuncionarioAdapter  extends BaseAdapter {
16      private Context context;
17      private ArrayList<Funcionario> funcionarios;
18
19      public FuncionarioAdapter(Context context, ArrayList<Funcionario> funcionarios) {
20          this.context = context;
21          this.funcionarios = funcionarios;
22      }
23
24      @Override
25      public int getCount() {
26          return funcionarios.size();
27      }
28
29      @Override
30      public Funcionario getItem(int position) {
31          return funcionarios.get(position);
32      }
33
34      @Override
35      public long getItemId(int position) {
36          return funcionarios.get(position).getId();
37      }
38
39      @Override
40      public View getView(int position, View root, ViewGroup parent) {
41          if (root == null) {
42              LayoutInflater vi = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
43              root = vi.inflate(R.layout.item_funcionario, parent, false);
44          }
45
46          TextView edtNome, edtCPF, edtTelefone;
```

```
        edtNome = (TextView) root.findViewById(R.id.tvNome);
        edtCPF = (TextView) root.findViewById(R.id.tvCPF);
        edtTelefone = (TextView) root.findViewById(R.id.tvTelefone);

        Funcionario funcionario = funcionarios.get(position);

        edtNome.setText(funcionario.getNome());
        edtCPF.setText(funcionario.getCpf());
        edtTelefone.setText(funcionario.getTelefone());

        return root;
    }
}
```

Código 44: FuncionarioAdapter.java

4.4.3 Item Adapter

```
package ds2.equipe1.restaurante.listas;

import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.EditText;
import android.widget.TextView;

import java.util.ArrayList;

import ds2.equipe1.restaurante.R;
import ds2.equipe1.restaurante.modelos.Item;

public class ItemAdapter extends BaseAdapter {
    private Context context;
    private ArrayList<Item> itens;

    public ItemAdapter(Context context, ArrayList<Item> fornecedores) {
        this.context = context;
        this.itens = fornecedores;
    }

    @Override
    public int getCount() {
        return itens.size();
    }

    @Override
    public Item getItem(int position) {
        return itens.get(position);
    }

    @Override
    public long getItemId(int position) {
        return itens.get(position).getId();
    }

    @Override
    public View getView(int position, View root, ViewGroup parent) {
        if (root == null) {
            LayoutInflater vi = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            root = vi.inflate(R.layout.item_itens, parent, false);
        }

        TextView edtNome,edtEstoque;
        edtNome = (TextView) root.findViewById(R.id.tvNome);
        edtEstoque = (TextView) root.findViewById(R.id.tvEstoque);

        Item item = itens.get(position);
        edtNome.setText(item.getNome());
        edtEstoque.setText(item.getQuantidade());

        return root;
    }
}
```

Código 45: ItemAdapter.java

4.4.4 Produto Adapter

```
package ds2.equipe1.restaurante.listas;

```

```java
import android.content.Context;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.TextView;

import java.util.ArrayList;

import ds2.equipe1.restaurante.R;
import ds2.equipe1.restaurante.modelos.Produto;

public class ProdutoAdapter extends BaseAdapter {
    private Context context;
    private ArrayList<Produto> produtos;

    public ProdutoAdapter(Context context, ArrayList<Produto> produtos) {
        this.context = context;
        this.produtos = produtos;
    }

    @Override
    public int getCount() {
        return produtos.size();
    }

    @Override
    public Produto getItem(int position) {
        return produtos.get(position);
    }

    @Override
    public long getItemId(int position) {
        return produtos.get(position).getId();
    }

    @Override
    public View getView(int position, View root, ViewGroup parent) {
        if (root == null) {
            LayoutInflater vi = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
            root = vi.inflate(R.layout.item_produto, parent, false);
        }

        TextView edtNome,edtPreco;
        edtNome = (TextView) root.findViewById(R.id.tvNome);
        edtPreco = (TextView) root.findViewById(R.id.tvPreco);

        Produto produto = produtos.get(position);

        edtNome.setText(produto.getNome());
        edtPreco.setText(Float.toString(produto.getPreco()));

        return root;
    }
}
```

Código 46: ProdutoAdapter.java

4.5 Helpers

4.5.1 Server Request

4.5.2 Request Callback

```java
package ds2.equipe1.restaurante.helpers;

import android.content.Context;

import java.util.ArrayList;

public abstract class RequestCallback<T> {
    private Context context;

    //cria request callback SEM tela de loading
    public RequestCallback(){
        onStart();
    }

    //Cria request callback COM uma tela de loading
    public RequestCallback(Context context){
        this.context = context;
        onStart();
    }

    public void onStart(){
```

```
22          if (context != null){
23              Utils.launchProgress(context);
24          }
25      }
26
27      public void execute(){
28          onFinish();
29      }
30      public void execute(ArrayList<T> lista) throws Exception {
31          onFinish();
32      }
33      public void execute(T object) throws  Exception {
34          onFinish();
35      }
36
37      public void onFinish(){
38          if (context != null) {
39              Utils.dismissProgress();
40          }
41      }
42  }
```

Código 47: 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  public class Utils {
17      public static final String TAG = "Restaurante";
18      private Context context;
19      private static ProgressDialog progress;
20
21      public Utils(Context context) {
22          this.context = context;
23      }
24
25      public void deleteData(String key){
26          SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
27          SharedPreferences.Editor e = settings.edit();
28          e.remove(key);
29          e.commit();
30      }
31
32      public float getData(String key, float defValue) {
33          SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
34          return settings.getFloat(key, defValue);
35      }
36
37      public String getData(String key, String defValue) {
38          SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
39          return settings.getString(key, defValue);
40      }
41
42      public int getData(String key, int defValue) {
43          SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
44          return settings.getInt(key, defValue);
45      }
46
47      public long getData(String key, long defValue) {
48          SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
49          return settings.getLong(key, defValue);
50      }
51
52      public boolean getData(String key, boolean defValue) {
53          SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
54          return settings.getBoolean(key, defValue);
55      }
56
57      public void setData(String key, float value) {
58          SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
59          settings.putFloat(key, value);
60          settings.commit();
61      }
62
```

```
63      public void setData(String key, String value) {
64          SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
65          settings.putString(key, value);
66          settings.commit();
67      }
68
69      public void setData(String key, int value) {
70          SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
71          settings.putInt(key, value);
72          settings.commit();
73      }
74
75      public void setData(String key, long value) {
76          SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
77          settings.putLong(key, value);
78          settings.commit();
79      }
80
81      public void setData(String key, boolean value) {
82          SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
83          settings.putBoolean(key, value);
84          settings.commit();
85      }
86
87      public void clearData(){
88          SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
89          SharedPreferences.Editor editor = settings.edit();
90          editor.clear();
91          editor.commit();
92      }
93
94      public static void launchProgress(Context context){
95          launchProgress(context, null, "Carregando...");
96      }
97
98      public static void launchProgress(Context context, String title, String text) {
99          launchProgress(context, title, text, null);
100     }
101
102     public static void launchProgress(Context context, String title, String text, DialogInterface.OnDismissListener
         func) {
103         progress = ProgressDialog.show(context, title, text, true);
104         progress.setCancelable(true);
105         if (func != null) progress.setOnDismissListener(func);
106     }
107
108     public static void dismissProgress() {
109         if (progress != null) {
110             progress.dismiss();
111         }
112     }
113
114     public void toast(String text){
115         Toast.makeText(context, text, Toast.LENGTH_LONG).show();
116     }
117
118     public void inputDialog(String title, String defValue, String hint, final DialogCallback callback){
119         AlertDialog.Builder builder = new AlertDialog.Builder(context);
120         builder.setTitle(title);
121
122         // Set up the input
123         LayoutInflater li = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
124         LinearLayout linearLayout = (LinearLayout) li.inflate(R.layout.input_dialog, null);
125         final EditText edtInput = (EditText) linearLayout.findViewById(R.id.edtInput);
126         edtInput.setHint(hint);
127         edtInput.setText(defValue);
128         builder.setView(linearLayout);
129
130         // Set up the buttons
131         builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
132             @Override
133             public void onClick(DialogInterface dialog, int which) {
134                 if (callback != null) {
135                     callback.execute(edtInput.getText().toString());
136                 }
137             }
138         });
139         builder.setNegativeButton("Cancelar", null);
140
141         builder.show();
142     }
143
144     public void selectPopup(String title, final DialogCallback callback){
145         AlertDialog.Builder builder = new AlertDialog.Builder(context);
146         builder.setTitle(title);
147
148         // Set up the input
149         LayoutInflater li = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
150         LinearLayout linearLayout = (LinearLayout) li.inflate(R.layout.select_popup, null);
151
152         final Spinner spinnerItens = (Spinner) linearLayout.findViewById(R.id.spinnerItens);
153         final EditText edtInput = (EditText) linearLayout.findViewById(R.id.edtInput);
154         builder.setView(linearLayout);
155
```

```
156         // Set up the buttons
157         builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
158             @Override
159             public void onClick(DialogInterface dialog, int which) {
160                 if (callback != null) {
161                     callback.execute(edtInput.getText().toString());
162                 }
163             }
164         });
165         builder.setNegativeButton("Cancelar", null);
166 
167         builder.show();
168     }
169 
170     public interface DialogCallback {
171         public void execute(String text);
172     }
173 }
```

Código 48: Utils.java

4.6 Banco de Dados

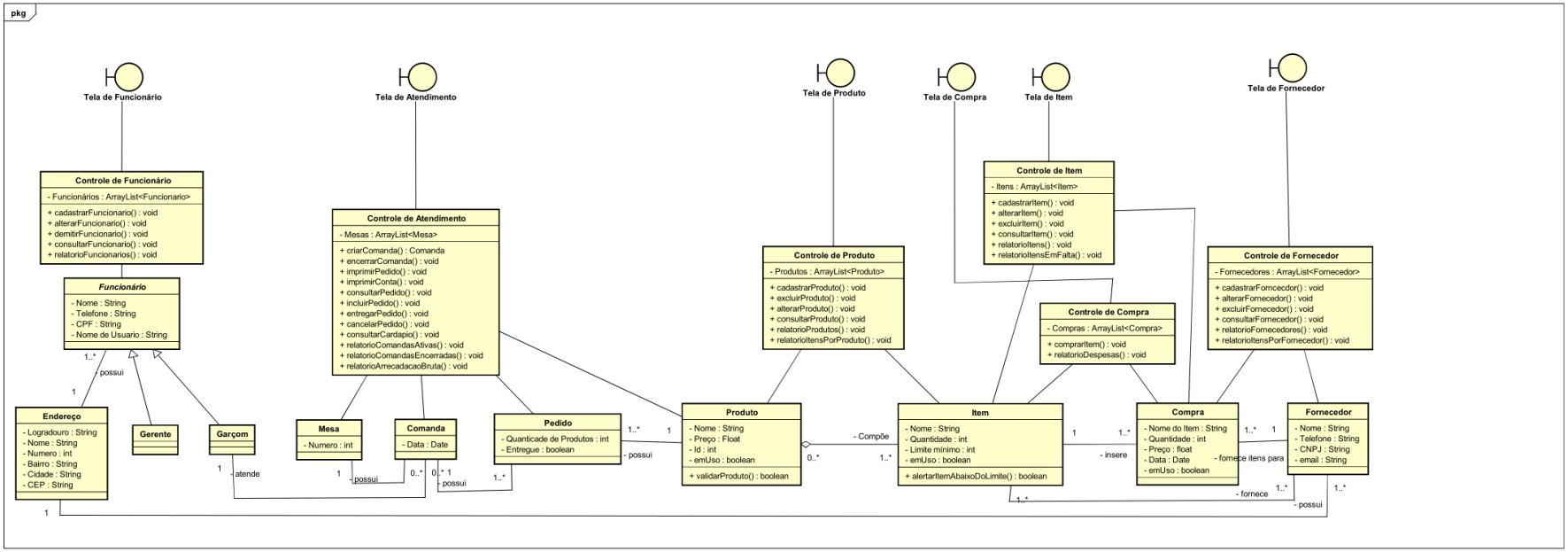
```
1  DROP TABLE endereco CASCADE;
2  DROP TABLE funcionario CASCADE;
3  DROP TABLE mesa CASCADE;
4  DROP TABLE comanda CASCADE;
5  DROP TABLE produto CASCADE;
6  DROP TABLE pedido CASCADE;
7  DROP TABLE item CASCADE;
8  DROP TABLE ingrediente CASCADE;
9  DROP TABLE fornecedor CASCADE;
10 DROP TABLE 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     entrege boolean,
63     id_produto integer,
64     id_comanda integer,
65     CONSTRAINT pk_pedido PRIMARY KEY (id),
```

```
    CONSTRAINT fk_pedido_comanda FOREIGN KEY (id_comanda) REFERENCES comanda (id),
    CONSTRAINT fk_pedido_produto FOREIGN KEY (id_produto) REFERENCES produto (id)
);

CREATE TABLE item (
    id serial,
    nome varchar(128),
    quantidade integer,
    limite integer,
    CONSTRAINT pk_item PRIMARY KEY (id)
);

CREATE TABLE ingrediente (
    id_item integer,
    id_produto integer,
    quantidade integer,
    CONSTRAINT pk_item_produto PRIMARY KEY (id_item, id_produto),
    CONSTRAINT fk_item FOREIGN KEY (id_item) REFERENCES item (id),
    CONSTRAINT fk_produto FOREIGN KEY (id_produto) REFERENCES produto (id)
);

CREATE TABLE fornecedor (
    id serial,
    nome varchar(128),
    telefone varchar(20),
    cnpj varchar(25),
    email varchar(128),
    id_endereco integer,
    CONSTRAINT pk_fornecedor PRIMARY KEY (id),
    CONSTRAINT fk_id_endereco FOREIGN KEY (id_endereco) REFERENCES endereco (id),
    CONSTRAINT unique_cnpj UNIQUE (cnpj)
);

CREATE TABLE compra (
    id serial,
    id_item integer,
    id_fornecedor integer,
    quantidade integer,
    preco float,
    data timestamp,
    CONSTRAINT pk_compra PRIMARY KEY (id),
    CONSTRAINT fk_compra_item FOREIGN KEY (id_item) REFERENCES item (id),
    CONSTRAINT fk_compra_fornecedor FOREIGN KEY (id_fornecedor) REFERENCES fornecedor (id)
);
```

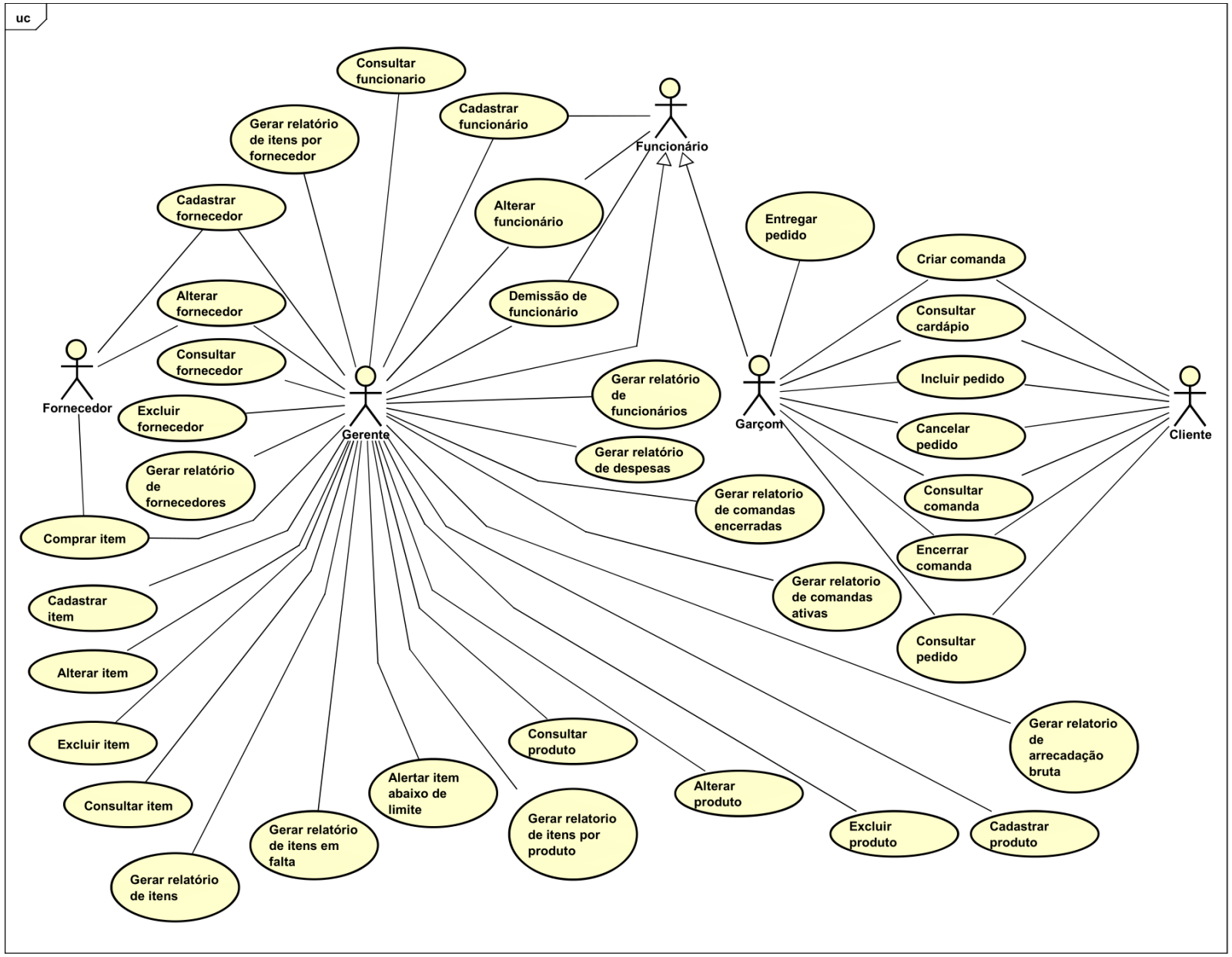
Código 49: CriarBanco.sql

5 Diagramas

5.1 Diagrama de Classes



5.2 Diagrama de Casos de Uso


uc
Consultar funcionario
Cadastrar funcionário
Gerar relatório de itens por fornecedor
Funcionário
Cadastrar fornecedor
Alterar funcionário
Entregar pedido
Criar comanda
Alterar fornecedor
Demissão de funcionário
Consultar cardápio
Consultar fornecedor
Gerar relatório de funcionários
Incluir pedido
Fornecedor
Gerente
Garçom
Cliente
Excluir fornecedor
Cancelar pedido
Gerar relatório de despesas
Gerar relatório de fornecedores
Consultar comanda
Gerar relatorio de comandas encerradas
Comprar item
Encerrar comanda
Gerar relatorio de comandas ativas
Cadastrar item
Consultar pedido
Alterar item
Excluir item
Gerar relatorio de arrecadação bruta
Consultar produto
Consultar item
Alertar item abaixo de limite
Alterar produto
Gerar relatório de itens em falta
Gerar relatorio de itens por produto
Excluir produto
Cadastrar produto
Gerar relatório de itens