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
FERNANDO MELO NASCIMENTO – 201210009310
FERNANDO 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

Devido ao avanço tecnológico e a crescente necessidade de encontrar soluções mais aprimoradas para o gerenciamento de um negócio, empresas buscam cada vez mais por sistemas que auxiliem no controle e gestão de suas atividades.

Esse projeto tem como objetivo a criação de um sistema para gerenciamento de restaurantes, permitindo o controle de diversas funções como: estoque, atendimento e gestão de funcionários. Esse sistema será integrado em computadores juntamente com o uso de tecnologias móveis de forma a permitir uma maior eficiência tanto no atendimento ao cliente como no controle das funções do restaurante. Devido a essa necessidade, além de uma rede de computadores, é necessário também que o estabelecimento disponha de dispositivos móveis (tablets ou smartphones) para o uso nas atividades.

2.1 Motivação

As atividades realizadas em um restaurante produzem uma grande quantidade de informações como: pedidos de clientes, compras com fornecedores, contratação de funcionários e gerenciamento de estoque. Devido a isso, surgem diversas dificuldades como:

- Gerência de grande volume de papel oriundo da venda de refeições, da compra de itens com fornecedores e da administração de funcionários.
- Lentidão no atendimento aos clientes, sendo esta uma das principais causas de cancelamento dos pedidos.
- Erros no preparo de pedidos dada a má compreensão do que é anotado em comandas.
- Erros de cálculo quando a conta de uma comanda é solicitada.

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 Codificação referente ao Cliente (Java/Android)

4.1 Modelos

4.1.1 Model

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

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

import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;

import org.json.JSONException;
import org.json.JSONObject;

import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;
import java.util.ArrayList;

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

public class Model<T> {
    protected Integer id;
    //transient para a serializacao nao incluir.
    protected transient Context context;

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

    public void setContext(Context context){
        this.context = context;
    }

    public void save(){
        save(null);
    }

    public void save(final RequestCallback<Model> callback){
        new ServerRequest(context).sendRequest(getControllerName(), ServerRequest.Action.SAVE, new Gson().toJson(this),
      new RequestCallback<JSONObject>() {
            @Override
            public void execute(JSONObject json) throws Exception {
                setId(json.getInt("id"));
                if (callback != null){
                    callback.execute(Model.this);
                }

                super.execute(json);
            }
        });
    }

    public void delete(){
        new ServerRequest(context).sendRequest(getControllerName(), ServerRequest.Action.DELETE, new Gson().toJson(this
     ), null);
        setId(null);
    }

    public static <T extends Model> void find(Context context, Class<T> klass, final Type typeArray, final
     RequestCallback<T> callback, int id){
        find(context, klass, typeArray, callback, id + "");
    }

    public static <T extends Model> void find(final Context context, Class<T> klass, final Type typeArray, final
     RequestCallback<T> callback, String where){
        ServerRequest serverRequest = new ServerRequest(context);
        String controller = ((Class<T>) ((ParameterizedType) klass.getGenericSuperclass()).getActualTypeArguments()[0])
     .getSimpleName();
        serverRequest.sendRequest(controller, ServerRequest.Action.FIND, where, new RequestCallback<JSONObject>() {
            @Override
            public void execute(JSONObject json) throws Exception {
                Log.w(Utils.TAG, "Trying cast to " + typeArray.toString());

                ArrayList<T> lista = new Gson().fromJson(json.getJSONArray("data").toString(), typeArray);
                for (T item : lista){
                    item.setContext(context);
                }
                if (callback != null){
                    if (lista.size() > 0) {
                        callback.execute(lista.get(0));
                    }
                    callback.execute(lista);
                }
                super.execute(json);
```

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

Código 1: Model.java

4.2 Aviso

```
1   package ds2.equipe1.restaurante.modelos;
2
3   import android.content.Context;
4
5   import java.util.Date;
6
7   public class Aviso extends Model<Aviso> {
8       private Item item;
9       private Date data;
10
11      public Aviso(Context context, Item item) {
12          super(context);
13          this.item = item;
14          this.data = new Date();
15      }
16  }
```

Código 2: Aviso.java

4.2.1 Comanda

```
1   package ds2.equipe1.restaurante.modelos;
2
3   import android.content.Context;
4
5   import java.util.ArrayList;
6   import java.util.Date;
7
8   public class Comanda extends Model<Comanda> {
9
10      private ArrayList<Pedido> pedidos = new ArrayList<>();
11      private String data;
12      private boolean ativa = false;
13      private String nome;
14      private int mesa;
15
16      public Comanda(Context context, String nome, int mesa, Pedido primeiro){
17          super(context);
18          Date agora = new Date();
19          pedidos.add(primeiro);
20          this.mesa = mesa;
21          data = agora.toString();
22          this.nome = nome;
23          ativa = true;
24      }
25
26      public void addPedido(Pedido outro){
27          pedidos.add(outro);
28      }
29
30      public void setNome(String nome){
31          this.nome=nome;
32      }
33
34      public ArrayList< Pedido > getPedidos(){
35          return pedidos;
```

```
36      }
37
38      public Pedido getPedido(int indice){
39          return pedidos.get(indice);
40      }
41
42      public void removerPedido(Pedido pedido){
43          pedidos.remove(pedido);
44      }
45
46      public boolean estaAtiva(){
47          return ativa;
48      }
49
50      public void desativar(){
51          ativa = false;
52      }
53
54      public float getCustoTotal(){
55          float custo = 0;
56          for (Pedido pedido : pedidos){
57              custo+= pedido.getCusto();
58          };
59          return custo;
60      }
61
62      public String getData() {
63          return data;
64      }
65
66      public int getMesa() {
67          return mesa;
68      }
69
70      public boolean isAtiva() {
71          return ativa;
72      }
73
74      public String getNome() {
75          return nome;
76      }
77  }
```

Código 3: Comanda.java

4.2.2 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; }
```

```
}
```

Código 4: Compra.java

4.2.3 Endereço

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

import android.content.Context;

public class Endereco extends Model<Endereco> {
    private String logradouro;
    private String rua;
    private int numero;
    private String bairro;
    private String cidade;
    private String estado;
    private String cep;

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

    public Endereco(Context context, String logradouro, String rua, int numero, String bairro, String cidade, String
     estado, String cep) {
        super(context);
        this.logradouro = logradouro;
        this.rua = rua;
        this.numero = numero;
        this.bairro = bairro;
        this.cidade = cidade;
        this.cep = cep;
        this.estado = estado;
    }

    public String getLogradouro() {
        return logradouro;
    }

    public String getRua() {
        return rua;
    }

    public int getNumero() {
        return numero;
    }

    public String getBairro() {
        return bairro;
    }

    public String getCidade() {
        return cidade;
    }

    public String getCep() {
        return cep;
    }

    public String getEstado() {
        return estado;
    }

    public void setLogradouro(String logradouro) {
        this.logradouro = logradouro;
    }

    public void setRua(String rua) {
        this.rua = rua;
    }

    public void setNumero(int numero) {
        this.numero = numero;
    }

    public void setBairro(String bairro) {
        this.bairro = bairro;
    }

    public void setCidade(String cidade) {
        this.cidade = cidade;
    }

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

    public void setCep(String cep) {
```

```
        this.cep = cep;
    }
}
```

Código 5: Endereco.jav

4.2.4 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 6: Fornecedor.java

4.2.5 Funcionário

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

import android.content.Context;

public class Funcionario extends Model<Funcionario> {
    public static final int GARCOM = 1;
```

```
7       public static final int GERENTE = 2;
8
9       private String nome;
10      private String telefone;
11      private String cpf;
12      private String nome_usuario;
13      private Integer tipo;
14      private Endereco endereco;
15
16      public Funcionario(Context context){
17          super(context);
18      }
19
20      public Funcionario(Context context, String nome, Endereco endereco, String telefone, String cpf, String
         nome_usuario, Integer tipo) {
21          super(context);
22          this.nome = nome;
23          this.telefone = telefone;
24          this.cpf = cpf;
25          this.endereco = endereco;
26          this.nome_usuario = nome_usuario;
27          this.tipo = tipo;
28      }
29
30      public String getNome() {
31          return nome;
32      }
33
34      public Endereco getEndereco() {
35          return endereco;
36      }
37
38      public String getTelefone() {
39          return telefone;
40      }
41
42      public String getCpf() {
43          return cpf;
44      }
45
46      public String getNome_usuario() {
47          return nome_usuario;
48      }
49
50      public Integer getTipo () { return tipo; };
51
52      public void setNome(String nome){
53          this.nome = nome;
54      }
55
56      public void setTelefone (String telefone){
57          this.telefone = telefone;
58      }
59
60      public void setCpf(String cnpj){
61          this.cpf = cnpj;
62      }
63
64      public void setEndereco(Endereco endereco) {
65          this.endereco = endereco;
66      }
67
68      public void setNome_usuario(String nome_usuario) { this.nome_usuario = nome_usuario; }
69
70      public void setTipo (Integer tipo) {this.tipo = tipo; };
71  }
```

Código 7: Funcionario.java

4.2.6 Garçom

```
1   package ds2.equipe1.restaurante.modelos;
2
3   import android.content.Context;
4
5   public class Garcom extends Funcionario {
6       public Garcom(Context context, String nome, Endereco endereco, String telefone, String cpf, String nome_de_usuario)
          {
7           super(context, nome, endereco, telefone, cpf, nome_de_usuario, GARCOM);
8       }
9   }
```

Código 8: Garcom.java

4.2.7 Gerente

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

import android.content.Context;

public class Gerente extends Funcionario {
    public Gerente(Context context, String nome, Endereco endereco, String telefone, String cpf, String nome_de_usuario
     ) {
        super(context, nome, endereco, telefone, cpf, nome_de_usuario, GERENTE);
    }
}
```

Código 9: Gerente.java

4.2.8 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 10: Ingrediente.jav

4.2.9 Item

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

import android.content.Context;

import com.google.gson.annotations.SerializedName;

import ds2.equipe1.restaurante.controles.ControleDeAtendimento;
import ds2.equipe1.restaurante.helpers.RequestCallback;

public class Item extends Model<Item> {
    private String nome;
    private String unidade;
    private int quantidade;
    @SerializedName("limite")
    private int limiteMinimo;

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

    public Item(Context context, String nome, int quantidade, String unidade, int limiteMinimo){
        super(context);
        this.nome = nome;
        this.unidade = unidade;
        this.quantidade = quantidade;
        this.limiteMinimo = limiteMinimo;
    }

    public void verificarItemAbaixoDoLimite(int quantidadeParaReduzir){
        if (getQuantidade()-quantidadeParaReduzir < getLimiteMinimo()){
            new Aviso(context, this).save();
        }
    }

    public String getNome(){
        return nome;
    }
```

```
38
39      public int getQuantidade(){
40          return quantidade;
41      }
42
43      public int getLimiteMinimo(){
44          return limiteMinimo;
45      }
46
47      public String getUnidade() {
48          return unidade;
49      }
50
51      public void setQuantidade(int quantidade){
52          //alterar quantidade no banco
53          this.quantidade = quantidade;
54      }
55
56      public void setLimiteMinimo(int limiteMinimo){
57          //alterar limiteMinimo no banco
58          this.limiteMinimo = limiteMinimo;
59      }
60
61      public void setNome(String nome){
62          this.nome = nome;
63      }
64
65      public void setUnidade(String unidade){
66          this.unidade=unidade;
67      }
68
69      @Override
70      public void save() {
71          unidade = unidade.toUpperCase();
72          super.save();
73      }
74
75      @Override
76      public void save(RequestCallback<Model> callback) {
77          unidade = unidade.toUpperCase();
78          super.save(callback);
79      }
80  }
```

Código 11: Item.java

4.2.10 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 int getNumero() {
37          return numero;
38      }
39
40      public Comanda getComanda(int indice){
```

```
41          return comandas.get(indice);
42      }
43  }
```

Código 12: Mesa.java

4.2.11 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 13: Pedido.java

4.2.12 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 = new ArrayList<>();
11
12      public Produto(Context context){
13          super(context);
14      }
15
16      public Produto(Context context, String nome, float preco, ArrayList<Ingrediente> ingredientes) {
17          super(context);
18          this.nome = nome;
19          this.preco = preco;
20          this.ingredientes = ingredientes;
21      }
22
23      public void setNome(String nome) {
24          this.nome = nome;
25      }
26
```

```
    /*public static ArrayList<Produto> carregarProdutos() {
        ArrayList<Produto> listaDeProdutos = new ArrayList<Produto>();

        ArrayList<Produto> lista = new ArrayList<Produto>();

        //conexao com o banco estabelecida
        for (int i = 0; i < lista.count(); i++) {
            listaDeProdutos.add(i, new Produto(lista.nome, lista.preco, lista.id, ...));
        }

        return listaDeProdutos; //(???)
    }*/

    public boolean validarProduto() {
        /*for (int i = 0; i < ingredientes.size(); i++) {
            if (ingredientes.get(i).getQuantidade() > ControleDeItem.consultarItem(ingredientes.get(i).getItem().
     getNome(), new RequestCallback<Item>() {
                @Override
                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;
    }

    public void setIngredientes(ArrayList<Ingrediente> ingredientes) {
        this.ingredientes = ingredientes;
    }

    //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 14: Produto.java

4.3 Controles

4.3.1 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);
```

```
21
22          //inserir comando sql pra saber numero de mesas
23          int numeroDeMesas = 20; //puxar numero do sql
24          for(int i = 1; i <= numeroDeMesas; i++) {
25              mesas.add(new Mesa(context, i));
26          }
27      }
28
29      public void criarComanda(Mesa mesa, Pedido pedido, String nome){
30          mesa.addComanda(new Comanda(context, nome, mesa.getNumero(), pedido));
31      }
32
33      public void encerrarComanda(Comanda comanda){
34          comanda.desativar();
35      }
36
37      public void imprimirPedido(Pedido pedido){
38          ArrayList<Produto> listaDeProdutos = pedido.getProdutos();
39          for(int i = 0; i < listaDeProdutos.size(); i++){
40              controleDeImpressao.imprimir(listaDeProdutos.get(i).getNome());
41          }
42      }
43
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 15: ControleDeAtendimento.java

4.3.2 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 16: ControleDeCompra.java

4.3.3 Controle de Fornecedor

```
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 ControleDeFornecedor {
17     //Lista com a consulta mais recente de fornecedores no servidor.
18     private ArrayList<Fornecedor> fornecedores = new ArrayList<>();
19     //Essa variavel serve para deixar uma referencia na memoria o tempo inteiro de qual esta selecionado,
20     //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).
21     private static Fornecedor selecionado;
22     private Context context;
23
24     public ControleDeFornecedor(Context context){
25         this.context = context;
26     }
27
28     public void salvarFornecedor(Fornecedor fornecedor, RequestCallback<Model> callback){
29         fornecedor.save(callback);
30     }
31
32     public boolean excluirFornecedor(Fornecedor fornecedor){
33         if (fornecedor.getId() != -1) {
34             fornecedor.delete();
35             return true;
36         } else return false;
37     }
38
39     public void consultarFornecedor(String consulta, final RequestCallback<Fornecedor> callback){
40         //Se a consulta for vazio, pega todos os itens do banco de dados, e coloca na memoria ram
41         if (consulta == null || fornecedores == null) {
42             Model.find(context, Fornecedor.class, new TypeToken<ArrayList<Fornecedor>>() {
43                     }.getType(), new RequestCallback<Fornecedor>() {
44                         @Override
45                         public void execute(ArrayList<Fornecedor> lista) throws Exception {
46                             super.execute(lista);
47
48                             fornecedores.clear();
49                             fornecedores.addAll(lista);
50
51                             if (callback != null){
52                                 callback.execute(lista);
53                             }
54                         }
55                     }, null);
56         } else {
57             consulta = consulta.toLowerCase();
58             //Se tiver consulta, faz a pesquisa nos itens que ja estao na memoria ram
59             try {
60                 ArrayList<Fornecedor> fornecedoresFiltrados = new ArrayList<>();
61
62                 for (Fornecedor fornecedor : fornecedores) {
63                     if (fornecedor.getNome().toLowerCase().contains(consulta) || fornecedor.getCnpj().contains(consulta
       )) {
64                         fornecedoresFiltrados.add(fornecedor);
65                     }
66                 }
67
68                 if (callback != null) {
69                     callback.execute(fornecedoresFiltrados);
70                 }
71             } catch (Exception e){
72                 Log.e(Utils.TAG, "Erro ao consultar fornecedores: ", e);
73             }
74         }
75     }
76
77     public void relatorioFornecedor(Fornecedor fornecedor){
78         //Banco de dados
79         //Exibir Relatorio
80     }
81
82     public void relatorioItensPorFornecedor(Item item, Fornecedor fornecedor){
83         //Banco de Dados
84         //Exibir Relatorio
85     }
86
87     public static void selecionarParaEditar(Fornecedor fornecedor){
88         ControleDeFornecedor.selecionado = fornecedor;
89     }
90
91     public static Fornecedor getSelecionado(){
92         return selecionado;
93     }
94
95     public static void deselecionar(){
96         selecionado = null;
97     }
98 }
```

Código 17: ControleDeFornecedor.java

4.3.4 Controle de Funcionário

```java
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;
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){
        this.context = context;
        controleDeImpressao = new ControleDeImpressao(context);
    }

    public void salvarFuncionario(Funcionario funcionario, RequestCallback<Model> callback){
        funcionario.save(callback);
    }

    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 == null || funcionarios == null) {
            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 {
            consulta = consulta.toLowerCase();
            //Se tiver consulta, faz a pesquisa nos itens que ja estao na memoria ram
            try {
                ArrayList<Funcionario> funcionariosFiltrados = new ArrayList<>();
```

```
81
82                  for (Funcionario funcionario : funcionarios) {
83                      if (funcionario.getNome().toLowerCase().contains(consulta) || funcionario.getCpf().contains(
        consulta)) {
84                          funcionariosFiltrados.add(funcionario);
85                      }
86                  }
87
88                  if (callback != null) {
89                      callback.execute(funcionariosFiltrados);
90                  }
91              } catch (Exception e){
92                  Log.e(Utils.TAG, "Erro ao consultar funcionarios: ", e);
93              }
94          }
95      }
96
97      public void relatorioFuncionario(Funcionario funcionario){
98          //Banco de dados
99          //Exibir Relatorio
100     }
101
102     public static void selecionarParaEditar(Funcionario funcionario){
103         ControleDeFuncionario.selecionado = funcionario;
104     }
105
106     public static Funcionario getSelecionado(){
107         return selecionado;
108     }
109
110     public static void deselecionar(){
111         selecionado = null;
112     }
113
114 }
```

Código 18: ControleDeFuncionario.java

4.3.5 Controle de Gerência

```
1  package ds2.equipe1.restaurante.controles;
2
3  public class ControleDeGerencia {
4      //ordenar avisos em ordem crescente de quantidade (urgencia), data meramente informativa
5
6  }
```

Código 19: ControleDeGerencia.java

4.3.6 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 20: ControleDeImpressao.java

4.3.7 Controle de Item

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

public class ControleDeItem {
    //Lista com a consulta mais recente de itens no servidor.
    private ArrayList<Item> itens = new ArrayList<>();
    private Context context;
    private static Item selecionado;

    private ControleDeImpressao controleDeImpressao;

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

    public void salvarItem(Item item) {
        item.save();
    }

    public void alterarItemQuantidade(Item item, int quantidade) {
        item.setQuantidade(quantidade);
        salvarItem(item);
    }

    public void alterarItemLimiteMinimo(Item item, int limiteMinimo) {
        item.setLimiteMinimo(limiteMinimo);
        salvarItem(item);
    }

    public void alterarItem(Item item, int quantidade, int limiteMinimo) {
        if (quantidade >= 0) {
            item.setQuantidade(quantidade);
        }

        if (limiteMinimo >= 0) {
            item.setLimiteMinimo(limiteMinimo);
        }

        salvarItem(item);
    }

    public void excluirItem(Item item) {
        item.delete();
    }

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

                    itens.clear();
                    itens.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<Item> itensFiltrados = new ArrayList<>();

                for (Item item : itens) {
                    if (item.getNome().contains(consulta)) {
                        itensFiltrados.add(item);
                    }
                }

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

```
94
95      public void relatorioItens(ArrayList<Item> itens) {
96          for (int i = 0; i < itens.size(); i++) {
97              Item item = itens.get(i);
98              // TODO: alterar para exibicao na tela do android ou gerar pdf
99              System.out.println(
100                     "Item: " + item.getNome() +
101                             ", Quantidade Disponivel: " + item.getQuantidade() +
102                             ", Limite Minimo: " + item.getLimiteMinimo()
103             );
104         }
105     }
106
107     public void relatorioItensEmFalta(ArrayList<Item> itens) {
108         for (int i = 0; i < itens.size(); i++) {
109             Item item = itens.get(i);
110             if (item.getQuantidade() < item.getLimiteMinimo()) {
111                 controleDeImpressao.imprimir(
112                         "Item em Falta: " + item.getNome() +
113                                 ", Quantidade Disponivel: " + item.getQuantidade() +
114                                 ", Limite Minimo: " + item.getLimiteMinimo()
115                 );
116             }
117         }
118     }
119
120     public static void selecionarParaEditar(Item item){
121         ControleDeItem.selecionado = item;
122     }
123
124     public static Item getSelecionado(){
125         return selecionado;
126     }
127
128     public static void deselecionar(){
129         selecionado = null;
130     }
131
132 }
```

Código 21: ControleDeItem.java

4.3.8 Controle de Produto

```
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.Produto;
13  import ds2.equipe1.restaurante.modelos.Item;
14  import ds2.equipe1.restaurante.modelos.Model;
15  import ds2.equipe1.restaurante.modelos.Produto;
16
17  public class ControleDeProduto {
18      //Lista com a consulta mais recente de produtos no servidor.
19      private ArrayList<Produto> produtos = new ArrayList<>();
20      //Essa variavel serve para deixar uma referencia na memoria o tempo inteiro de qual esta selecionado,
21      //para que quando haja uma edicao, a alteracao seja refletida em mais de uma tela (exemplo: editar um produto e
        atualizar na tela de busca).
22      private static Produto selecionado;
23      private Context context;
24
25      public ControleDeProduto(Context context){
26          this.context = context;
27      }
28
29      public void salvarProduto(Produto produto, RequestCallback<Model> callback){
30          produto.save(callback);
31      }
32
33      public boolean excluirProduto(Produto produto){
34          if (produto.getId() != -1) {
35              produto.delete();
36              return true;
37          } else return false;
38      }
39
40      public void consultarProduto(String consulta, final RequestCallback<Produto> callback){
41          //Se a consulta for vazio, pega todos os itens do banco de dados, e coloca na memoria ram
42          if (consulta == null || produtos == null) {
43              Model.find(context, Produto.class, new TypeToken<ArrayList<Produto>>() {
```

```
            }.getType(), new RequestCallback<Produto>() {
                @Override
                public void execute(ArrayList<Produto> lista) throws Exception {
                    super.execute(lista);

                    produtos.clear();
                    produtos.addAll(lista);

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

                for (Produto produto : produtos) {
                    if (produto.getNome().toLowerCase().contains(consulta)) {
                        produtosFiltrados.add(produto);
                    }
                }

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

    public void relatorioProduto(Produto produto){
        //Banco de dados
        //Exibir Relatorio
    }

    public void relatorioItensPorProduto(Item item, Produto produto){
        //Banco de Dados
        //Exibir Relatorio
    }

    public static void selecionarParaEditar(Produto produto){
        ControleDeProduto.selecionado = produto;
    }

    public static Produto getSelecionado(){
        return selecionado;
    }

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

Código 22: ControleDeProduto.java

4.4 Fronteiras

4.4.1 Main Activity

```
package ds2.equipe1.restaurante;

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

public class MainActivity extends AppCompatActivity {

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

        //TODO: Tela de log in e botao de log off
    }
}
```

Código 23: MainActivity.java

4.4.2 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 24: MenuPrincipal.java

4.4.3 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 25: MenuFornecedores.java

4.4.4 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) {
```

```
                open(CadastroFuncionario.class);
            }
        });
    }

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

Código 26: MenuFuncionarios.java

4.4.5 Menu Garçom

```
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.Toast;

public class MenuGarcom extends AppCompatActivity {

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

        findViewById(R.id.menu_criar_comanda).setOnClickListener(new View.OnClickListener() {
            @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);
            }
        });

        findViewById(R.id.menu_comandas_ativas).setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Toast.makeText(MenuGarcom.this, "Falta␣criar␣tela␣e␣adicionar␣Requisito␣Funcional,␣Diagramas␣etc",
     Toast.LENGTH_LONG).show();
            }
        });
    }

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

Código 27: MenuGarcom.java

4.4.6 Menu Itens

```
package ds2.equipe1.restaurante;

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

public class MenuItens extends AppCompatActivity {

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

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

```

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

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

Código 28: MenuItens.java

4.4.7 Menu Produtos

```
package ds2.equipe1.restaurante;

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

public class MenuProdutos extends AppCompatActivity {

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

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

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

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

Código 29: MenuProdutos.java

4.4.8 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 30: MenuRelatorio.java

4.4.9 Busca Fornecedor

```
package ds2.equipe1.restaurante;

import android.content.Intent;
import android.os.Bundle;
```

```java
import android.support.v7.app.AppCompatActivity;
import android.view.Menu;
import android.view.View;
import android.widget.AdapterView;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.ListView;

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(null);
    }

    private void init(){
        lvFornecedores = (ListView) findViewById(R.id.lvFornecedores);

        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) {
                ControleDeFornecedor.selecionarParaEditar(adapter.getItem(position));

                Intent intent = new Intent(BuscaFornecedor.this, CadastroFornecedor.class);
                intent.putExtra("alterar", true);
                startActivity(intent);
            }
        });
    }

    private void consultar(final String consulta){
        controleDeFornecedor.consultarFornecedor(consulta, new RequestCallback<Fornecedor>(this) {
            @Override
            public void execute(ArrayList<Fornecedor> fornecedores) throws Exception {
                super.execute(fornecedores);

                BuscaFornecedor.this.fornecedores.clear();
                BuscaFornecedor.this.fornecedores.addAll(fornecedores);

                adapter.notifyDataSetChanged();
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.busca, menu);
        Utils.prepararSearchMenu(this, menu, new Utils.DialogCallback() {
            @Override
            public void execute(String text) {
                consultar(text);
            }
        });
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    protected void onResume() {
        //verificar se o fornecedor foi excluido para remover da tela.
        for (int i = 0; i < fornecedores.size(); i++){
            if (fornecedores.get(i).getId() == null){
                fornecedores.remove(i--);
            }
        }

```

```
99          //Quando a tela reabrir, atualizar a interface com as informacoes alteradas do item selecionado.
100         adapter.notifyDataSetChanged();
101
102         super.onResume();
103     }
104 }
```

Código 31: BuscaFornecedor.java

4.4.10 Busca Funcionário

```
1   package ds2.equipe1.restaurante;
2
3   import android.content.Intent;
4   import android.os.Bundle;
5   import android.support.v7.app.AppCompatActivity;
6   import android.view.Menu;
7   import android.view.View;
8   import android.widget.AdapterView;
9   import android.widget.ListView;
10
11  import java.util.ArrayList;
12
13  import ds2.equipe1.restaurante.controles.ControleDeFuncionario;
14  import ds2.equipe1.restaurante.helpers.RequestCallback;
15  import ds2.equipe1.restaurante.helpers.Utils;
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
26      @Override
27      protected void onCreate(Bundle savedInstanceState) {
28          super.onCreate(savedInstanceState);
29          setContentView(R.layout.activity_busca_funcionario);
30
31          init();
32
33          controleDeFuncionario = new ControleDeFuncionario(this);
34
35          funcionarios = new ArrayList<>();
36          adapter = new FuncionarioAdapter(BuscaFuncionario.this, funcionarios);
37          lvFuncionarios.setAdapter(adapter);
38
39          consultar(null);
40      }
41      private void init(){
42          //pegar referencias dos componentes xml
43          lvFuncionarios = (ListView) findViewById(R.id.lvFuncionarios);
44
45          lvFuncionarios.setOnItemClickListener(new AdapterView.OnItemClickListener() {
46              @Override
47              //no clique do item o android nos da a view, a posicao do item na lista e o ID do item (que nos
     configuramos)
48              //entao passamos esse ID para
49              public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
50                  ControleDeFuncionario.selecionarParaEditar(funcionarios.get(position));
51
52                  Intent intent = new Intent(BuscaFuncionario.this, CadastroFuncionario.class);
53                  intent.putExtra("alterar", true);
54                  startActivity(intent);
55              }
56          });
57      }
58
59      private void consultar(final String consulta){
60          controleDeFuncionario.consultarFuncionario(consulta, new RequestCallback<Funcionario>(this) {
61              @Override
62              public void execute(ArrayList<Funcionario> funcionarios) throws Exception {
63                  super.execute(funcionarios);
64
65                  BuscaFuncionario.this.funcionarios.clear();
66                  BuscaFuncionario.this.funcionarios.addAll(funcionarios);
67
68                  adapter.notifyDataSetChanged();
69              }
70          });
71      }
72
73      @Override
74      public boolean onCreateOptionsMenu(Menu menu) {
75          getMenuInflater().inflate(R.menu.busca, menu);
76          Utils.prepararSearchMenu(this, menu, new Utils.DialogCallback() {
```

```
77              @Override
78              public void execute(String text) {
79                  consultar(text);
80              }
81          });
82          return super.onCreateOptionsMenu(menu);
83      }
84
85      @Override
86      protected void onResume() {
87          //verificar se o Funcionario foi excluido para remover da tela.
88          for (int i = 0; i < funcionarios.size(); i++){
89              if (funcionarios.get(i).getId() == null){
90                  funcionarios.remove(i--);
91              }
92          }
93
94          //Quando a tela reabrir, atualizar a interface com as informacoes alteradas do item selecionado.
95          adapter.notifyDataSetChanged();
96
97          super.onResume();
98      }
99
100
101
102     public void onItemClick(View v){
103         startActivity(new Intent(this, CadastroFuncionario.class));
104     }
105
106
107 }
```

Código 32: BuscaFuncionario.java

4.4.11 Busca Item

```
1  package ds2.equipe1.restaurante;
2
3  import android.content.Intent;
4  import android.os.Bundle;
5  import android.support.v7.app.AppCompatActivity;
6  import android.view.Menu;
7  import android.view.View;
8  import android.widget.AdapterView;
9  import android.widget.ListView;
10
11 import java.util.ArrayList;
12
13 import ds2.equipe1.restaurante.controles.ControleDeItem;
14 import ds2.equipe1.restaurante.helpers.RequestCallback;
15 import ds2.equipe1.restaurante.helpers.Utils;
16 import ds2.equipe1.restaurante.listas.ItemAdapter;
17 import ds2.equipe1.restaurante.modelos.Item;
18
19
20 public class BuscaItem extends AppCompatActivity {
21
22     private ControleDeItem controleDeItem;
23
24     private ListView lvItens;
25     private ItemAdapter adapter;
26
27     private ArrayList<Item> itens;
28
29     @Override
30     protected void onCreate(Bundle savedInstanceState) {
31         super.onCreate(savedInstanceState);
32         setContentView(R.layout.activity_busca_fornecedor);
33
34         init();
35
36         controleDeItem = new ControleDeItem(this);
37
38         itens = new ArrayList<>();
39         adapter = new ItemAdapter(BuscaItem.this, itens);
40         lvItens.setAdapter(adapter);
41
42         consultar(null);
43     }
44
45     private void init(){
46         //pegar referencias dos componentes xml
47         lvItens = (ListView) findViewById(R.id.lvFornecedores);
48
49         lvItens.setOnItemClickListener(new AdapterView.OnItemClickListener() {
50             @Override
51             public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
52                 ControleDeItem.selecionarParaEditar(adapter.getItem(position));
```

```
                Intent intent = new Intent(BuscaItem.this, CadastroItem.class);
                intent.putExtra("alterar", true);
                startActivity(intent);
            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.busca, menu);
        Utils.prepararSearchMenu(this, menu, new Utils.DialogCallback() {
            @Override
            public void execute(String text) {
                consultar(text);
            }
        });
        return super.onCreateOptionsMenu(menu);
    }

    private void consultar(String consulta){
        controleDeItem.consultarItem(consulta, new RequestCallback<Item>(){
            @Override
            public void execute(ArrayList<Item> itens) throws Exception {
                BuscaItem.this.itens.clear();
                BuscaItem.this.itens.addAll(itens);
                adapter.notifyDataSetChanged();

                super.execute(itens);
            }
        });
    }
}
```

Código 33: BuscaItem.java

4.4.12 Busca Produto

```
package ds2.equipe1.restaurante;

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

import java.util.ArrayList;

import ds2.equipe1.restaurante.controles.ControleDeProduto;
import ds2.equipe1.restaurante.helpers.RequestCallback;
import ds2.equipe1.restaurante.helpers.Utils;
import ds2.equipe1.restaurante.listas.ProdutoAdapter;
import ds2.equipe1.restaurante.modelos.Produto;

public class BuscaProduto extends AppCompatActivity {
    private ControleDeProduto controleDeProduto;

    private ListView lvProdutos;
    private ProdutoAdapter adapter;
    //Produtos visiveis na tela
    private ArrayList<Produto> produtos;
    private EditText edtProcurar;
    private ImageView ivProcurar;

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

        init();

        controleDeProduto = new ControleDeProduto(this);

        produtos = new ArrayList<>();
        adapter = new ProdutoAdapter(BuscaProduto.this, produtos);
        lvProdutos.setAdapter(adapter);

        consultar(null);
    }
    private void init(){
        lvProdutos = (ListView) findViewById(R.id.lvProdutos);

        lvProdutos.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
```

```
52              // (que nos configuramos) entao passamos esse ID para
53              public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
54                  ControleDeProduto.selecionarParaEditar(adapter.getItem(position));
55
56                  Intent intent = new Intent(BuscaProduto.this, CadastroProduto.class);
57                  intent.putExtra("alterar", true);
58                  startActivity(intent);
59              }
60          });
61      }
62
63      private void consultar(final String consulta){
64          controleDeProduto.consultarProduto(consulta, new RequestCallback<Produto>(this) {
65              @Override
66              public void execute(ArrayList<Produto> produtos) throws Exception {
67                  super.execute(produtos);
68
69                  BuscaProduto.this.produtos.clear();
70                  BuscaProduto.this.produtos.addAll(produtos);
71
72                  adapter.notifyDataSetChanged();
73              }
74          });
75      }
76
77      @Override
78      public boolean onCreateOptionsMenu(Menu menu) {
79          getMenuInflater().inflate(R.menu.busca, menu);
80          Utils.prepararSearchMenu(this, menu, new Utils.DialogCallback() {
81              @Override
82              public void execute(String text) {
83                  consultar(text);
84              }
85          });
86          return super.onCreateOptionsMenu(menu);
87      }
88
89      @Override
90      protected void onResume() {
91          //verificar se o produto foi excluido para remover da tela.
92          for (int i = 0; i < produtos.size(); i++){
93              if (produtos.get(i).getId() == null){
94                  produtos.remove(i--);
95              }
96          }
97
98          //Quando a tela reabrir, atualizar a interface com as informacoes alteradas do item selecionado.
99          adapter.notifyDataSetChanged();
100
101         super.onResume();
102     }
103 }
```

Código 34: BuscaProduto.java

4.4.13 Cadastro Endereço

```
1  package ds2.equipe1.restaurante;
2
3  //(edt.*?)\.setText\(endereco.get(.*?)\(\)\);
4  //endereco.set$2($1.getText().toString());
5
6  import android.content.Intent;
7  import android.os.Bundle;
8  import android.support.v7.app.AppCompatActivity;
9  import android.view.View;
10 import android.widget.Button;
11 import android.widget.EditText;
12
13 import ds2.equipe1.restaurante.controles.ControleDeEndereco;
14 import ds2.equipe1.restaurante.modelos.Endereco;
15
16 public class CadastroEndereco extends AppCompatActivity {
17     private EditText edtCEP, edtLogradouro, edtRua, edtBairro, edtCidade, edtEstado, edtNumero;
18     private Button btnCadastrar;
19     private Endereco endereco;
20
21     @Override
22     protected void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.activity_cadastro_endereco);
25
26         init();
27
28         Intent intent = getIntent();
29
30         if (intent.getBooleanExtra("alterar", false)){
31             carregarEndereco();
```

```
32              btnCadastrar.setText("Alterar");
33          } else {
34              endereco = new Endereco(this);
35              ControleDeEndereco.selecionarParaEditar(endereco);
36          }
37      }
38
39      private void init(){
40          btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
41          edtRua = (EditText) findViewById(R.id.edtRua);
42          edtCEP = (EditText) findViewById(R.id.edtCEP);
43          edtLogradouro = (EditText) findViewById(R.id.edtLogradouro);
44          edtBairro = (EditText) findViewById(R.id.edtBairro);
45          edtCidade = (EditText) findViewById(R.id.edtCidade);
46          edtEstado = (EditText) findViewById(R.id.edtEstado);
47          edtNumero = (EditText) findViewById(R.id.edtNumero);
48
49          btnCadastrar.setOnClickListener(new View.OnClickListener() {
50              @Override
51              public void onClick(View v) {
52
53                  endereco.setLogradouro(edtLogradouro.getText().toString());
54                  endereco.setRua(edtRua.getText().toString());
55                  endereco.setCep(edtCEP.getText().toString());
56                  endereco.setLogradouro(edtLogradouro.getText().toString());
57                  endereco.setBairro(edtBairro.getText().toString());
58                  endereco.setCidade(edtCidade.getText().toString());
59                  endereco.setEstado(edtEstado.getText().toString());
60                  endereco.setNumero(Integer.parseInt(edtNumero.getText().toString()));
61
62                  Intent i = getIntent();
63                  setResult(RESULT_OK, i);
64                  finish();
65              }
66          });
67      }
68
69      private void carregarEndereco(){
70          endereco = ControleDeEndereco.getSelecionado();
71          if (endereco != null){
72              edtLogradouro.setText(endereco.getLogradouro());
73              edtRua.setText(endereco.getRua());
74              edtCEP.setText(endereco.getCep());
75              edtLogradouro.setText(endereco.getLogradouro());
76              edtBairro.setText(endereco.getBairro());
77              edtCidade.setText(endereco.getCidade());
78              edtEstado.setText(endereco.getEstado());
79              edtNumero.setText(String.valueOf(endereco.getNumero()));
80          }
81      }
82  }
```

Código 35: CadastroEndereco.java

4.4.14 Cadastro Fornecedor

```
1   package ds2.equipe1.restaurante;
2
3   import android.content.Intent;
4   import android.os.Bundle;
5   import android.support.v7.app.AppCompatActivity;
6   import android.view.View;
7   import android.widget.Button;
8   import android.widget.EditText;
9   import android.widget.Toast;
10
11  import ds2.equipe1.restaurante.controles.ControleDeEndereco;
12  import ds2.equipe1.restaurante.controles.ControleDeFornecedor;
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.Model;
17
18  public class CadastroFornecedor extends AppCompatActivity {
19      private ControleDeFornecedor controleDeFornecedor;
20      private EditText edtNome, edtCNPJ, edtEndereco, edtEmail, edtTelefone;
21      private Button btnCadastrar, btnCadastrarEndereco, btnExcluir;
22
23      private Fornecedor fornecedor;
24      private boolean novoCadastro = true;
25
26      @Override
27      protected void onCreate(Bundle savedInstanceState) {
28          super.onCreate(savedInstanceState);
29          setContentView(R.layout.activity_cadastro_fornecedor);
30
31          init();
32
```

```
33          fornecedor = new Fornecedor(this);
34          controleDeFornecedor = new ControleDeFornecedor(this);
35
36          Intent intent = getIntent();
37          if (intent.getBooleanExtra("alterar", false)){
38              novoCadastro = false;
39              carregarFornecedor();
40          }
41
42          if (novoCadastro){
43              btnExcluir.setVisibility(View.GONE);
44          } else {
45              btnCadastrar.setText("Alterar");
46              btnExcluir.setVisibility(View.VISIBLE);
47          }
48      }
49
50      private void init(){
51          edtNome = (EditText) findViewById(R.id.edtNome);
52          edtCNPJ = (EditText) findViewById(R.id.edtCNPJ);
53          edtEndereco = (EditText) findViewById(R.id.edtEndereco);
54          edtEmail = (EditText) findViewById(R.id.edtEmail);
55          edtTelefone = (EditText) findViewById(R.id.edtTelefone);
56          btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
57          btnCadastrarEndereco = (Button) findViewById(R.id.btnCadastrarEndereco);
58          btnExcluir = (Button) findViewById(R.id.btnExcluir);
59
60          btnCadastrarEndereco.setOnClickListener(new View.OnClickListener() {
61              @Override
62              public void onClick(View v) {
63                  onCadastrarEnderecoClick();
64              }
65          });
66
67          btnExcluir.setOnClickListener(new View.OnClickListener() {
68              @Override
69              public void onClick(View v) {
70                  if (!novoCadastro && fornecedor.getId() != null) {
71                      fornecedor.delete();
72                      fornecedor.setId(null);
73                      new Utils(CadastroFornecedor.this).toast("Fornecedor␣excluido!");
74                      finish();
75                  }
76              }
77          });
78
79          btnCadastrar.setOnClickListener(new View.OnClickListener() {
80              @Override
81              public void onClick(View v) {
82                  onCadastrarClick();
83              }
84          });
85      }
86
87      private void onCadastrarEnderecoClick(){
88          Intent intent = new Intent(this, CadastroEndereco.class);
89          if (fornecedor.getEndereco() != null) {
90              ControleDeEndereco.selecionarParaEditar(fornecedor.getEndereco());
91              intent.putExtra("alterar", true);
92          }
93          startActivityForResult(intent,1);
94      }
95
96      private void onCadastrarClick(){
97          final String nome = edtNome.getText().toString();
98          final String CNPJ = edtCNPJ.getText().toString();
99          final String email = edtEmail.getText().toString();
100         final String telefone = edtTelefone.getText().toString();
101
102         if (fornecedor.getEndereco() == null || nome.isEmpty() || CNPJ.isEmpty() || email.isEmpty() || telefone.isEmpty
        ()){
103             Toast.makeText(CadastroFornecedor.this, "Necessario␣todos␣os␣campos", Toast.LENGTH_SHORT).show();
104             return;
105         }
106
107         fornecedor.setNome(nome.trim());
108         fornecedor.setCnpj(CNPJ.trim());
109         fornecedor.setEmail(email.trim());
110         fornecedor.setTelefone(telefone.replaceAll("␣", ""));
111         controleDeFornecedor.salvarFornecedor(fornecedor, null);
112
113         if (fornecedor.getId() == null) {
114             new Utils(CadastroFornecedor.this).toast("Fornecedor␣cadastrado!");
115         } else {
116             new Utils(CadastroFornecedor.this).toast("Fornecedor␣alterado!");
117         }
118
119         ControleDeFornecedor.deselecionar();
120         ControleDeEndereco.deselecionar();
121
122         finish();
123     }
124
125     public void carregarFornecedor(){
```

```
126         if (ControleDeFornecedor.getSelecionado() != null) {
127             CadastroFornecedor.this.fornecedor = ControleDeFornecedor.getSelecionado();
128 
129             edtNome.setText(fornecedor.getNome());
130             edtCNPJ.setText(fornecedor.getCnpj());
131             edtTelefone.setText(fornecedor.getTelefone());
132             edtEmail.setText(fornecedor.getEmail());
133             if (fornecedor.getEndereco() != null) {
134                 btnCadastrarEndereco.setText("Alterar");
135                 edtEndereco.setText(fornecedor.getEndereco().getRua());
136             } else {
137                 btnCadastrarEndereco.setText("Cadastrar");
138             }
139         }
140     }
141 
142     @Override
143     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
144         super.onActivityResult(requestCode, resultCode, data);
145 
146         if (requestCode == 1 && resultCode == RESULT_OK){
147             fornecedor.setEndereco(ControleDeEndereco.getSelecionado());
148             edtEndereco.setText(fornecedor.getEndereco().getRua());
149             btnCadastrarEndereco.setText("Alterar");
150         }
151     }
152 }
```

Código 36: CadastroFornecedor.java

4.4.15 Cadastro 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.ArrayAdapter;
8  import android.widget.Button;
9  import android.widget.EditText;
10 import android.widget.Spinner;
11 import android.widget.Toast;
12 
13 import ds2.equipe1.restaurante.controles.ControleDeEndereco;
14 import ds2.equipe1.restaurante.controles.ControleDeFuncionario;
15 import ds2.equipe1.restaurante.helpers.Utils;
16 import ds2.equipe1.restaurante.modelos.Funcionario;
17 
18 public class CadastroFuncionario extends AppCompatActivity {
19 
20     private ControleDeFuncionario controleDeFuncionario;
21     private EditText edtNome, edtCPF, edtEndereco, edtTelefone, edtNome_de_usuario;
22     private Spinner spTipo;
23 
24     private Button btnCadastrar, btnCadastrarEndereco, btnExcluir;
25 
26     private Funcionario funcionario;
27     private boolean novoCadastro = true;
28     private String[] arraySpinner;
29 
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_cadastro_funcionario);
34 
35         init();
36 
37         funcionario = new Funcionario(this);
38         controleDeFuncionario = new ControleDeFuncionario(this);
39 
40         Intent intent = getIntent();
41         if (intent.getBooleanExtra("alterar", false)){
42             novoCadastro = false;
43             carregarFuncionario();
44         }
45 
46         if (novoCadastro){
47             btnExcluir.setVisibility(View.GONE);
48         } else {
49             btnCadastrar.setText("Alterar");
50             btnExcluir.setVisibility(View.VISIBLE);
51         }
52     }
53 
54     private void init(){
55         edtNome = (EditText) findViewById(R.id.edtNome);
56         edtCPF = (EditText) findViewById(R.id.edtCPF);
```

```
57          edtNome_de_usuario = (EditText) findViewById(R.id.edtNome_de_usuario);
58          edtEndereco = (EditText) findViewById(R.id.edtEndereco);
59          edtTelefone = (EditText) findViewById(R.id.edtTelefone);
60          spTipo = (Spinner) findViewById(R.id.edtTipo);
61
62          btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
63          btnCadastrarEndereco = (Button) findViewById(R.id.btnCadastrarEndereco);
64          btnExcluir = (Button) findViewById(R.id.btnExcluir);
65
66          this.arraySpinner = new String[]{
67                  "Garcom", "Gerente"
68          };
69
70          ArrayAdapter<String> adapter = new ArrayAdapter<String>(this,
71                  android.R.layout.simple_spinner_item, arraySpinner);
72          adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
73          spTipo.setAdapter(adapter);
74
75          btnCadastrarEndereco.setOnClickListener(new View.OnClickListener() {
76              @Override
77              public void onClick(View v) {
78                  onCadastrarEnderecoClick();
79              }
80          });
81
82          btnExcluir.setOnClickListener(new View.OnClickListener() {
83              @Override
84              public void onClick(View v) {
85                  if (!novoCadastro && funcionario.getId() != null) {
86                      funcionario.delete();
87                      funcionario.setId(null);
88                      new Utils(CadastroFuncionario.this).toast("Funcionario excluido!");
89                      finish();
90                  }
91              }
92          });
93
94          btnCadastrar.setOnClickListener(new View.OnClickListener() {
95              @Override
96              public void onClick(View v) {
97                  onCadastrarClick();
98              }
99          });
100     }
101
102     private void onCadastrarEnderecoClick(){
103         Intent intent = new Intent(this, CadastroEndereco.class);
104         if (funcionario.getEndereco() != null) {
105             ControleDeEndereco.selecionarParaEditar(funcionario.getEndereco());
106             intent.putExtra("alterar", true);
107         }
108         startActivityForResult(intent,1);
109     }
110
111     private void onCadastrarClick(){
112         final String nome = edtNome.getText().toString();
113         final String CPF = edtCPF.getText().toString();
114         final String telefone = edtTelefone.getText().toString();
115         final String nome_de_usuario = edtNome_de_usuario.getText().toString();
116
117         if (funcionario.getEndereco() == null || nome.isEmpty() || CPF.isEmpty() || nome_de_usuario.isEmpty() ||
     telefone.isEmpty()){
118             Toast.makeText(CadastroFuncionario.this, "Necessario todos os campos", Toast.LENGTH_SHORT).show();
119             return;
120         }
121         final int tipo = spTipo.getSelectedItemPosition() + 1;
122
123         funcionario.setNome(nome.trim());
124         funcionario.setCpf(CPF.trim());
125         funcionario.setNome_usuario(nome_de_usuario.trim());
126         funcionario.setTelefone(telefone.replaceAll(" ", ""));
127         funcionario.setTipo(tipo);
128         controleDeFuncionario.salvarFuncionario(funcionario, null);
129
130         if (funcionario.getId() == null) {
131             new Utils(this).toast("Funcionario cadastrado!");
132         } else {
133             new Utils(this).toast("Funcionario alterado!");
134         }
135
136         ControleDeFuncionario.deselecionar();
137         ControleDeEndereco.deselecionar();
138         finish();
139     }
140
141     public void carregarFuncionario(){
142         if (ControleDeFuncionario.getSelecionado() != null) {
143             CadastroFuncionario.this.funcionario = ControleDeFuncionario.getSelecionado();
144
145             edtNome.setText(funcionario.getNome());
146             edtCPF.setText(funcionario.getCpf());
147             edtNome_de_usuario.setText(funcionario.getNome_usuario().toLowerCase());
148             edtTelefone.setText(funcionario.getTelefone());
149             if (funcionario.getEndereco() != null) {
```

```
150                 btnCadastrarEndereco.setText("Alterar");
151                 edtEndereco.setText(funcionario.getEndereco().getRua());
152             } else {
153                 btnCadastrarEndereco.setText("Cadastrar");
154             }
155         }
156     }
157
158     @Override
159     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
160         super.onActivityResult(requestCode, resultCode, data);
161
162         if (requestCode == 1 && resultCode == RESULT_OK){
163             funcionario.setEndereco(ControleDeEndereco.getSelecionado());
164             edtEndereco.setText(funcionario.getEndereco().getRua());
165             btnCadastrarEndereco.setText("Alterar");
166         }
167     }
168 }
```

Código 37: CadastroFuncionario.java

4.4.16 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 import android.widget.Toast;
11
12 import ds2.equipe1.restaurante.controles.ControleDeItem;
13 import ds2.equipe1.restaurante.helpers.Utils;
14 import ds2.equipe1.restaurante.modelos.Item;
15
16 public class CadastroItem extends AppCompatActivity {
17
18     private EditText edtNome,edtQuantidade,edtUnidade,edtLimiteMinimo;
19     private Button btnCadastrar, btnExcluir;
20     private ControleDeItem controleDeItem;
21     private Item item;
22
23     private boolean novoCadastro = true;
24
25     @Override
26     protected void onCreate(Bundle savedInstanceState) {
27         super.onCreate(savedInstanceState);
28         setContentView(R.layout.activity_cadastro_item);
29
30         init();
31
32         item = new Item(this);
33         controleDeItem = new ControleDeItem(this);
34
35         Intent intent = getIntent();
36         if (intent.getBooleanExtra("alterar", false)){
37             novoCadastro = false;
38             carregarItem();
39         }
40
41         if (novoCadastro){
42             btnExcluir.setVisibility(View.GONE);
43         } else {
44             btnCadastrar.setText("Alterar");
45             btnExcluir.setVisibility(View.VISIBLE);
46         }
47     }
48
49     private void init(){
50         edtNome = (EditText) findViewById(R.id.edtNome);
51         edtQuantidade = (EditText) findViewById(R.id.edtQuantidade);
52         edtUnidade = (EditText) findViewById(R.id.edtUnidade);
53         edtLimiteMinimo = (EditText) findViewById(R.id.edtLimiteMinimo);
54         btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
55         btnExcluir = (Button) findViewById(R.id.btnExcluir);
56
57         btnExcluir.setOnClickListener(new View.OnClickListener() {
58             @Override
59             public void onClick(View v) {
60                 if (!novoCadastro && item.getId() != null) {
61                     item.delete();
62                     item.setId(null);
63                     new Utils(CadastroItem.this).toast("Item␣excluido!");
64                     finish();
```

```
                }
            }
        });

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


    private void onCadastrarClick() {
        String nome = edtNome.getText().toString();
        String unidade = edtUnidade.getText().toString();
        int quantidade;
        try {
            quantidade = Integer.parseInt(edtQuantidade.getText().toString());
        } catch (Exception e){
            quantidade = 0;
        }
        int limite;
        try {
            limite = Integer.parseInt(edtLimiteMinimo.getText().toString());
        } catch (Exception e){
            limite = 0;
        }

        if (nome.isEmpty() || unidade.isEmpty()){
            Toast.makeText(CadastroItem.this, "Necessario todos os campos", Toast.LENGTH_SHORT).show();
            return;
        }

        item.setNome(nome.trim());
        item.setQuantidade(quantidade);
        item.setUnidade(unidade.trim());
        item.setLimiteMinimo(limite);

        controleDeItem.salvarItem(item);

        if (item.getId() == null) {
            new Utils(this).toast("Item cadastrado!");
        } else {
            new Utils(this).toast("Item alterado!");
        }
        finish();
    }

    private void carregarItem(){
        if (ControleDeItem.getSelecionado() != null) {
            CadastroItem.this.item = ControleDeItem.getSelecionado();

            edtNome.setText(item.getNome());
            edtQuantidade.setText(""+item.getQuantidade());
            edtUnidade.setText(item.getUnidade());
            edtLimiteMinimo.setText(""+item.getLimiteMinimo());
        }
    }


}
```

Código 38: CadastroItem.java

4.4.17 Cadastro Produto

```
package ds2.equipe1.restaurante;

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

import java.util.ArrayList;

import ds2.equipe1.restaurante.controles.ControleDeFornecedor;
import ds2.equipe1.restaurante.controles.ControleDeItem;
import ds2.equipe1.restaurante.controles.ControleDeProduto;
import ds2.equipe1.restaurante.helpers.RequestCallback;
import ds2.equipe1.restaurante.helpers.Utils;
import ds2.equipe1.restaurante.listas.IngredienteAdapter;
import ds2.equipe1.restaurante.modelos.Ingrediente;
```

```
22  import ds2.equipe1.restaurante.modelos.Item;
23  import ds2.equipe1.restaurante.modelos.Produto;
24
25  public class CadastroProduto extends AppCompatActivity {
26
27      private ControleDeProduto controleDeProduto;
28      private ControleDeItem controleDeItem;
29      private EditText edtNome, edtPreco;
30      private ListView lvIngredientes;
31      private IngredienteAdapter adapter;
32      private Button btnCadastrar, btnExcluir;
33      private ImageButton btnAddIngrediente;
34      private ArrayList<Item> itens = new ArrayList<>(); //apenas para o dropdown
35      private ArrayList<Ingrediente> ingredientes = new ArrayList<>();
36      private Produto produto;
37
38      @Override
39      protected void onCreate(Bundle savedInstanceState) {
40          super.onCreate(savedInstanceState);
41          setContentView(R.layout.activity_cadastro_produto);
42
43          init();
44
45          produto = new Produto(this);
46
47          controleDeProduto = new ControleDeProduto(this);
48          controleDeItem = new ControleDeItem(this);
49
50          controleDeItem.consultarItem(null, new RequestCallback<Item>() {
51              @Override
52              public void execute(ArrayList<Item> lista) throws Exception {
53                  itens.clear();
54                  itens.addAll(lista);
55                  super.execute(lista);
56              }
57          });
58
59
60          boolean novoCadastro = true;
61          Intent intent = getIntent();
62          if (intent.getBooleanExtra("alterar", false)){
63              novoCadastro = false;
64              carregarProduto();
65          }
66
67          if (novoCadastro){
68              btnExcluir.setVisibility(View.GONE);
69          } else {
70              btnCadastrar.setText("Alterar");
71              btnExcluir.setVisibility(View.VISIBLE);
72          }
73      }
74
75      private void init(){
76          edtNome = (EditText) findViewById(R.id.edtNome);
77          edtPreco = (EditText) findViewById(R.id.edtPreco);
78          lvIngredientes = (ListView) findViewById(R.id.lvIngredientes);
79          btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
80          btnExcluir = (Button) findViewById(R.id.btnExcluir);
81          //btnAddIngrediente = (ImageButton) findViewById(R.id.btnAddIngrediente);
82
83          adapter = new IngredienteAdapter(this, ingredientes);
84          lvIngredientes.setAdapter(adapter);
85
86          btnCadastrar.setOnClickListener(new View.OnClickListener() {
87              @Override
88              public void onClick(View v) {
89                  onCadastrarClick();
90              }
91          });
92
93          btnExcluir.setOnClickListener(new View.OnClickListener() {
94              @Override
95              public void onClick(View v) {
96                  onCancelarClick();
97              }
98          });
99      }
100
101     private void onCadastrarClick(){
102         final String nome = edtNome.getText().toString();
103         final Float preco = Float.parseFloat(edtPreco.getText().toString());
104
105         if (nome.isEmpty()){
106             Toast.makeText(CadastroProduto.this, "Necessario␣todos␣os␣campos", Toast.LENGTH_SHORT).show();
107             return;
108         }
109
110         produto.setNome(nome);
111         produto.setPreco(preco);
112         produto.setIngredientes(ingredientes);
113
114         produto.save();
115
```

```
116         new Utils(this).toast("Produto cadastrado!");
117         finish();
118     }
119
120     private void onCancelarClick(){
121         finish();
122     }
123
124     public void onAddIngredienteClick(){
125         new Utils(this).selectPopup("Cadastrar ingrediente", new Utils.IngredienteCallback() {
126             @Override
127             public void execute(Item item, int quantidade) {
128                 ingredientes.add(new Ingrediente(CadastroProduto.this, item, quantidade));
129                 adapter.notifyDataSetChanged();
130             }
131         }, itens);
132     }
133
134     private void carregarProduto(){
135         if (ControleDeProduto.getSelecionado() != null) {
136             produto = ControleDeProduto.getSelecionado();
137
138             edtNome.setText(produto.getNome());
139             edtPreco.setText(""+produto.getPreco());
140             ingredientes.clear();
141             ingredientes.addAll(produto.getIngredientes());
142             adapter.notifyDataSetChanged();
143         }
144     }
145 }
```

Código 39: CadastroProduto.java

4.4.18 Criação Comanda

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

```
54          // as you specify a parent activity in AndroidManifest.xml.
55          int id = item.getItemId();
56
57          if (id == android.R.id.home) {
58              onBackPressed();
59              return true;
60          }
61
62          return super.onOptionsItemSelected(item);
63      }
64
65      public void onItemClick(View v){
66          startActivity(new Intent(this, ExibirPedido.class));
67      }
68  }
```

Código 40: CriacaoComanda.java

4.4.19 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 41: ExibirPedido.java

4.4.20 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 42: SelecionarIngredientes.java

4.4.21 Tela Garçom

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

```
28      }
29
30      public void open(Class activity){
31          startActivity(new Intent(this, activity));
32      }
33  }
```

Código 43: TelaGarcom.java

4.4.22 Tela Relatórios

```
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.widget.TextView;
7
8   public class TelaRelatorios extends AppCompatActivity {
9       private TextView tvImpressora;
10
11      @Override
12      protected void onCreate(Bundle savedInstanceState) {
13          super.onCreate(savedInstanceState);
14          setContentView(R.layout.activity_tela_relatorios);
15          init();
16
17          Intent i = getIntent();
18
19          if (i.hasExtra("dados")){
20              tvImpressora.setText(i.getStringExtra("dados"));
21          } else {
22              tvImpressora.setText("Nenhum␣dado␣passado␣como␣parametro");
23          }
24      }
25
26      private void init(){
27          tvImpressora = (TextView) findViewById(R.id.tvImpressora);
28      }
29  }
```

Código 44: TelaRelatorios.java

4.5 Adapters

4.5.1 Fornecedor Adapter

```
1   package ds2.equipe1.restaurante.listas;
2
3   import android.content.Context;
4   import android.view.LayoutInflater;
5   import android.view.View;
6   import android.view.ViewGroup;
7   import android.widget.BaseAdapter;
8   import android.widget.EditText;
9   import android.widget.TextView;
10
11  import java.util.ArrayList;
12
13  import ds2.equipe1.restaurante.R;
14  import ds2.equipe1.restaurante.helpers.Utils;
15  import ds2.equipe1.restaurante.modelos.Fornecedor;
16
17  public class FornecedorAdapter extends BaseAdapter {
18      private Context context;
19      private ArrayList<Fornecedor> fornecedores;
20
21      public FornecedorAdapter(Context context, ArrayList<Fornecedor> fornecedores) {
22          this.context = context;
23          this.fornecedores = fornecedores;
24      }
25
26      @Override
27      public int getCount() {
28          return fornecedores.size();
29      }
30
31      @Override
32      public Fornecedor getItem(int position) {
33          return fornecedores.get(position);
34      }
```

```
35
36      @Override
37      public long getItemId(int position) {
38          return position;
39      }
40
41      @Override
42      public View getView(int position, View root, ViewGroup parent) {
43          if (root == null) {
44              LayoutInflater vi = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
45              root = vi.inflate(R.layout.item_fornecedor, parent, false);
46          }
47
48          TextView edtNome, edtCNPJ, edtTelefone;
49          edtNome = (TextView) root.findViewById(R.id.tvNome);
50          edtCNPJ = (TextView) root.findViewById(R.id.tvCNPJ);
51          edtTelefone = (TextView) root.findViewById(R.id.tvTelefone);
52
53          Fornecedor fornecedor = fornecedores.get(position);
54
55          edtNome.setText(fornecedor.getNome());
56          edtCNPJ.setText(fornecedor.getCnpj());
57          edtTelefone.setText(fornecedor.getTelefone());
58
59          new Utils(context).addFuncaoRemover(this, root.findViewById(R.id.ivRemover), fornecedores, position);
60
61          return root;
62      }
63  }
```

Código 45: FornecedorAdapter.java

4.5.2 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.helpers.Utils;
14  import ds2.equipe1.restaurante.modelos.Funcionario;
15
16  public class FuncionarioAdapter  extends BaseAdapter {
17      private Context context;
18      private ArrayList<Funcionario> funcionarios;
19
20      public FuncionarioAdapter(Context context, ArrayList<Funcionario> funcionarios) {
21          this.context = context;
22          this.funcionarios = funcionarios;
23      }
24
25      @Override
26      public int getCount() {
27          return funcionarios.size();
28      }
29
30      @Override
31      public Funcionario getItem(int position) {
32          return funcionarios.get(position);
33      }
34
35      @Override
36      public long getItemId(int position) {
37          return position;
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_funcionario, parent, false);
45          }
46
47          TextView edtNome, edtCPF, edtTelefone;
48          edtNome = (TextView) root.findViewById(R.id.tvNome);
49          edtCPF = (TextView) root.findViewById(R.id.tvCPF);
50          edtTelefone = (TextView) root.findViewById(R.id.tvTelefone);
51
52          Funcionario funcionario = funcionarios.get(position);
53
54          edtNome.setText(funcionario.getNome());
```

```
        edtCPF.setText(funcionario.getCpf());
        edtTelefone.setText(funcionario.getTelefone());

        new Utils(context).addFuncaoRemover(this, root.findViewById(R.id.ivRemover), funcionarios, position);

        return root;
    }
}
```

Código 46: FuncionarioAdapter.java

4.5.3 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.TextView;

import java.util.ArrayList;

import ds2.equipe1.restaurante.R;
import ds2.equipe1.restaurante.helpers.Utils;
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 position;
    }

    @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_item, 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() + item.getUnidade());

        new Utils(context).addFuncaoRemover(this, root.findViewById(R.id.ivRemover), itens, position);

        return root;
    }
}
```

Código 47: ItemAdapter.java

4.5.4 Produto Adapter

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

import android.content.Context;
import android.view.LayoutInflater;
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.helpers.Utils;
14 import ds2.equipe1.restaurante.modelos.Produto;
15
16 public class ProdutoAdapter extends BaseAdapter {
17     private Context context;
18     private ArrayList<Produto> produtos;
19
20     public ProdutoAdapter(Context context, ArrayList<Produto> produtos) {
21         this.context = context;
22         this.produtos = produtos;
23     }
24
25     @Override
26     public int getCount() {
27         return produtos.size();
28     }
29
30     @Override
31     public Produto getItem(int position) {
32         return produtos.get(position);
33     }
34
35     @Override
36     public long getItemId(int position) {
37         return position;
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_produto, parent, false);
45         }
46
47         TextView edtNome,edtPreco;
48         edtNome = (TextView) root.findViewById(R.id.tvNome);
49         edtPreco = (TextView) root.findViewById(R.id.tvPreco);
50
51         Produto produto = produtos.get(position);
52
53         edtNome.setText(produto.getNome());
54         edtPreco.setText(Float.toString(produto.getPreco()));
55
56         new Utils(context).addFuncaoRemover(this, root.findViewById(R.id.ivRemover), produtos, position);
57
58         return root;
59     }
60 }
```

Código 48: ProdutoAdapter.java

4.6 Helpers

4.6.1 Server Request

```
1  package ds2.equipe1.restaurante.helpers;
2
3  import android.content.Context;
4  import android.content.Intent;
5  import android.util.Log;
6  import android.widget.Toast;
7
8  import com.androidquery.AQuery;
9  import com.androidquery.callback.AjaxCallback;
10 import com.androidquery.callback.AjaxStatus;
11 import com.google.gson.Gson;
12 import com.google.gson.internal.Excluder;
13 import com.google.gson.reflect.TypeToken;
14 import com.google.gson.stream.MalformedJsonException;
15
16 import org.json.JSONArray;
17 import org.json.JSONException;
18 import org.json.JSONObject;
19
20 import java.lang.reflect.Type;
21 import java.util.ArrayList;
22 import java.util.HashMap;
23 import java.util.Map;
24 import java.util.Objects;
```

```
25
26  import ds2.equipe1.restaurante.modelos.Fornecedor;
27  import ds2.equipe1.restaurante.modelos.Ingrediente;
28  import ds2.equipe1.restaurante.modelos.Model;
29  import ds2.equipe1.restaurante.modelos.Produto;
30
31  /**
32   * Created by Fernando on 07/04/2016.
33   */
34  public class ServerRequest {
35      private static final String TAG = Utils.TAG;
36
37      public enum Action {
38          INDEX, SAVE, DELETE, GET, FIND, LOGIN
39      }
40
41      private Context context;
42
43      public ServerRequest(Context context) {
44          this.context = context;
45      }
46
47      public void sendRequest(final String controller, Action action, String data, final RequestCallback<JSONObject>
         callback){
48          Map<String, Object> params = getAuthParams();
49          params.put("data", data);
50
51          new AQuery(context).ajax(buildURL(controller, action), params, String.class, new AjaxCallback<String>(){
52              @Override
53              public void callback(String url, String str, AjaxStatus status) {
54                  try {
55                      Log.w(TAG, "[" + url + "]:");
56                      JSONObject json = new JSONObject(str);
57                      Log.w(TAG, json.toString());
58                      if (json.getBoolean("success")) {
59                          if (callback != null) {
60                              callback.execute(json);
61                          }
62                      } else {
63                          String errors = "";
64                          if (json.has("errors")) {
65                              JSONArray msgs = json.getJSONArray("errors");
66                              for (int i = 0; i < msgs.length(); i++){
67                                  errors += "␣" + msgs.getString(i);
68                              }
69                          } else {
70                              errors = "Erro␣desconhecido␣no␣servidor";
71                          }
72                          Toast.makeText(context, errors, Toast.LENGTH_SHORT).show();
73                      }
74                  } catch (Exception e){
75                      Toast.makeText(context, "Falha␣ao␣processar␣resposta:\n\n" + e.getLocalizedMessage(), Toast.
       LENGTH_SHORT).show();
76                      Log.e(TAG, "Erro␣no␣processamento␣de␣dados:␣", e);
77                      if (str != null)
78                          Log.e(TAG, str);
79                      Log.e(TAG, "AjaxStatus:␣" + status.getError() + "␣---␣" + status.getCode() + "␣---␣" + status.
       getMessage());
80                      if (status.getMessage().contains("network")){
81                          Toast.makeText(context, "Falha␣na␣conexao␣com␣" + url, Toast.LENGTH_LONG).show();
82                      }
83                  }
84                  super.callback(url, str, status);
85              }
86
87              @Override
88              public void failure(int code, String message) {
89                  Log.e(TAG, "Falha␣de␣conexao:␣" + message + "␣" + code);
90                  super.failure(code, message);
91              }
92          }.timeout(5000));
93      }
94
95      private Map<String, Object> getAuthParams(){
96          Map<String, Object> params = new HashMap<>();
97          //TODO: Remover padrao
98          params.put("usuario", new Utils(context).getData("usuario", "user.default"));
99          params.put("senha", new Utils(context).getData("senha", "senha.default"));
100         return params;
101     }
102
103     private String buildURL(String controller, Action action){
104         return new Utils(context).getData("host", "http://179.232.16.215:8080/") + controller.toLowerCase() + "/" +
        action.name().toLowerCase() + "/";
105     }
106 }
```

Código 49: ServerRequest.java

4.6.2 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(){
        if (context != null){
            Utils.launchProgress(context);
        }
    }

    public void execute(){
        onFinish();
    }
    public void execute(ArrayList<T> lista) throws Exception {
        onFinish();
    }
    public void execute(T object) throws  Exception {
        onFinish();
    }

    public void onFinish(){
        if (context != null) {
            Utils.dismissProgress();
        }
    }
}
```

Código 50: RequestCallback.java

4.6.3 Utilitários

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

import android.app.Activity;
import android.app.ProgressDialog;
import android.app.SearchManager;
import android.content.Context;
import android.content.DialogInterface;
import android.content.SharedPreferences;
import android.support.v7.app.AlertDialog;
import android.support.v7.widget.SearchView;
import android.text.InputType;
import android.view.LayoutInflater;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.BaseAdapter;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.Spinner;
import android.widget.Toast;

import java.util.ArrayList;
import java.util.List;

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

public class Utils {
    public static final String TAG = "Restaurante";
    private Context context;
    private static ProgressDialog progress;

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

```

```
39      public void deleteData(String key){
40          SharedPreferences settings = context.getSharedPreferences(TAG, 0);
41          SharedPreferences.Editor e = settings.edit();
42          e.remove(key);
43          e.commit();
44      }
45
46      public float getData(String key, float defValue) {
47          SharedPreferences settings = context.getSharedPreferences(TAG, 0);
48          return settings.getFloat(key, defValue);
49      }
50
51      public String getData(String key, String defValue) {
52          SharedPreferences settings = context.getSharedPreferences(TAG, 0);
53          return settings.getString(key, defValue);
54      }
55
56      public int getData(String key, int defValue) {
57          SharedPreferences settings = context.getSharedPreferences(TAG, 0);
58          return settings.getInt(key, defValue);
59      }
60
61      public long getData(String key, long defValue) {
62          SharedPreferences settings = context.getSharedPreferences(TAG, 0);
63          return settings.getLong(key, defValue);
64      }
65
66      public boolean getData(String key, boolean defValue) {
67          SharedPreferences settings = context.getSharedPreferences(TAG, 0);
68          return settings.getBoolean(key, defValue);
69      }
70
71      public void setData(String key, float value) {
72          SharedPreferences.Editor settings = context.getSharedPreferences(TAG, 0).edit();
73          settings.putFloat(key, value);
74          settings.commit();
75      }
76
77      public void setData(String key, String value) {
78          SharedPreferences.Editor settings = context.getSharedPreferences(TAG, 0).edit();
79          settings.putString(key, value);
80          settings.commit();
81      }
82
83      public void setData(String key, int value) {
84          SharedPreferences.Editor settings = context.getSharedPreferences(TAG, 0).edit();
85          settings.putInt(key, value);
86          settings.commit();
87      }
88
89      public void setData(String key, long value) {
90          SharedPreferences.Editor settings = context.getSharedPreferences(TAG, 0).edit();
91          settings.putLong(key, value);
92          settings.commit();
93      }
94
95      public void setData(String key, boolean value) {
96          SharedPreferences.Editor settings = context.getSharedPreferences(TAG, 0).edit();
97          settings.putBoolean(key, value);
98          settings.commit();
99      }
100
101     public void clearData(){
102         SharedPreferences settings = context.getSharedPreferences(TAG, 0);
103         SharedPreferences.Editor editor = settings.edit();
104         editor.clear();
105         editor.commit();
106     }
107
108     public static void launchProgress(Context context){
109         launchProgress(context, null, "Carregando...");
110     }
111
112     public static void launchProgress(Context context, String title, String text) {
113         launchProgress(context, title, text, null);
114     }
115
116     public static void launchProgress(Context context, String title, String text, DialogInterface.OnDismissListener
        func) {
117         progress = ProgressDialog.show(context, title, text, true);
118         progress.setCancelable(true);
119         if (func != null) progress.setOnDismissListener(func);
120     }
121
122     public static void dismissProgress() {
123         if (progress != null) {
124             progress.dismiss();
125         }
126     }
127
128     public void toast(String text){
129         Toast.makeText(context, text, Toast.LENGTH_LONG).show();
130     }
131
```

```
132     public void inputDialog(String title, String defValue, String hint, final DialogCallback callback){
133         AlertDialog.Builder builder = new AlertDialog.Builder(context);
134         builder.setTitle(title);
135 
136         // Set up the input
137         LayoutInflater li = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
138         LinearLayout linearLayout = (LinearLayout) li.inflate(R.layout.input_dialog, null);
139         final EditText edtInput = (EditText) linearLayout.findViewById(R.id.edtInput);
140         edtInput.setHint(hint);
141         edtInput.setText(defValue);
142         builder.setView(linearLayout);
143 
144         // Set up the buttons
145         builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
146             @Override
147             public void onClick(DialogInterface dialog, int which) {
148                 if (callback != null) {
149                     callback.execute(edtInput.getText().toString());
150                 }
151             }
152         });
153         builder.setNegativeButton("Cancelar", null);
154 
155         builder.show();
156     }
157 
158     public void selectPopup(String title, final IngredienteCallback callback, final ArrayList<Item> itens){
159         AlertDialog.Builder builder = new AlertDialog.Builder(context);
160         builder.setTitle(title);
161 
162         // Set up the input
163         LayoutInflater li = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
164         LinearLayout linearLayout = (LinearLayout) li.inflate(R.layout.select_popup, null);
165 
166         final Spinner spinnerItens = (Spinner) linearLayout.findViewById(R.id.spinnerItens);
167         List<String> spinnerArray =  new ArrayList<>();
168         for (Item item : itens) {
169             spinnerArray.add(item.getNome());
170         }
171 
172         ArrayAdapter<String> adapter = new ArrayAdapter<>(
173                 context, android.R.layout.simple_spinner_item, spinnerArray);
174 
175         adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
176         spinnerItens.setAdapter(adapter);
177 
178         final EditText edtInput = (EditText) linearLayout.findViewById(R.id.edtInput);
179         edtInput.setInputType(InputType.TYPE_CLASS_NUMBER);
180         builder.setView(linearLayout);
181 
182         // Set up the buttons
183         builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
184             @Override
185             public void onClick(DialogInterface dialog, int which) {
186                 if (callback != null) {
187                     callback.execute(itens.get(spinnerItens.getSelectedItemPosition()), Integer.parseInt(edtInput.
     getText().toString()));
188                 }
189             }
190         });
191         builder.setNegativeButton("Cancelar", null);
192 
193         builder.show();
194     }
195 
196     public void simNaoDialog(String titulo, String mensagem, final DialogCallback callbackSim, final DialogCallback
     callbackNao){
197         AlertDialog.Builder builder = new AlertDialog.Builder(context);
198         builder.setTitle(titulo);
199         builder.setMessage(mensagem);
200 
201         // Set up the buttons
202         builder.setPositiveButton("Sim", new DialogInterface.OnClickListener() {
203             @Override
204             public void onClick(DialogInterface dialog, int which) {
205                 if (callbackSim != null) {
206                     callbackSim.execute("");
207                 }
208             }
209         });
210         builder.setNegativeButton("Nao", new DialogInterface.OnClickListener() {
211             @Override
212             public void onClick(DialogInterface dialog, int which) {
213                 if (callbackNao != null){
214                     callbackNao.execute("");
215                 }
216             }
217         });
218 
219         builder.show();
220     }
221 
222     public interface IngredienteCallback {
223         public void execute(Item item, int quantidade);
```

```
224     }
225
226     public interface DialogCallback {
227         public void execute(String text);
228     }
229
230     public static void prepararSearchMenu(Activity activity, Menu menu, final DialogCallback dialogCallback){
231         MenuItem searchItem = menu.findItem(R.id.action_search);
232
233         SearchManager searchManager = (SearchManager) activity.getSystemService(Context.SEARCH_SERVICE);
234
235         SearchView searchView = null;
236         if (searchItem != null) {
237             searchView = (SearchView) searchItem.getActionView();
238         }
239
240         if (searchView != null) {
241             searchView.setSearchableInfo(searchManager.getSearchableInfo(activity.getComponentName()));
242             SearchView.OnQueryTextListener queryTextListener = new SearchView.OnQueryTextListener() {
243                 public boolean onQueryTextChange(String search) {
244                     dialogCallback.execute(search);
245                     return true;
246                 }
247
248                 public boolean onQueryTextSubmit(String query) {
249                     return false;
250                 }
251             };
252
253             searchView.setOnQueryTextListener(queryTextListener);
254         }
255     }
256
257     public <T> void addFuncaoRemover(final BaseAdapter adapter, View view, final ArrayList<T> arr, final int position){
258         if (view != null)
259             view.setOnClickListener(new View.OnClickListener() {
260                 @Override
261                 public void onClick(View v) {
262                     simNaoDialog("Excluir", "Deseja␣excluir␣este␣item?", new DialogCallback() {
263                         @Override
264                         public void execute(String text) {
265                             Model model = (Model) arr.get(position);
266                             model.delete();
267                             arr.remove(position);
268                             adapter.notifyDataSetChanged();
269                         }
270                     }, null);
271                 }
272             });
273     }
274 }
```

Código 51: Utils.java

5 Codificação referente ao Servidor (PHP)

5.1 Modelos

5.1.1 Comanda

```php
<?php

use Phalcon\Mvc\Model;
use Phalcon\Mvc\Model\Message;

//TODO: Remover total de pedidos
class Comanda extends Model {

    public $id;
    public $data;
    public $ativa;
    public $nome;
    public $pedidos;

    public function getSchema()  {
        return "public";
    }
}

?>
```

Código 52: Comanda.php

5.1.2 Compra

```php
<?php

use Phalcon\Mvc\Model;
use Phalcon\Mvc\Model\Message;


class Compra extends Model {

    public $id;
    public $item;
    public $quantidade;
    public $preco;
    public $data;

    public function getSchema()  {
        return "public";
    }
}

?>
```

Código 53: Compra.php

5.1.3 Endereço

```php
<?php

use Phalcon\Mvc\Model;
use Phalcon\Mvc\Model\Message;


class Endereco extends Model {

    public $id;
    public $logradouro;
    public $rua;
    public $numero;
    public $bairro;
    public $cidade;
    public $estado;
    public $cep;

    /*public function beforeSave(){
        if (!$rua || !$logradouro || !$bairro){
            return false;
        }
    }*/
```

```
23
24      public function initialize(){
25          $this->hasMany("id", "Fornecedor", "id_endereco");
26          $this->hasMany("id", "Funcionario", "id_endereco");
27      }
28
29      public function getSchema()  {
30          return "public";
31      }
32  }
33
34  ?>
```

Código 54: Endereco.php

5.1.4 Fornecedor

```
1   <?php
2
3   //http://localhost/restaurante/fornecedor/save/?data={%22nome%22:%22Fernando%20Messias%22,%20%22telefone
        %22:%20%2279%2098833848%22,%20%22cnpj%22:%20%2200-000-000/0000-00%22,%20%22email%22:%22fernandoxlr@live.com
        %22,%20%22id_endereco%22:1}
4
5   use Phalcon\Mvc\Model;
6   use Phalcon\Mvc\Model\Message;
7
8
9   class Fornecedor extends Model {
10
11      public $id;
12      public $nome;
13      public $telefone;
14      public $cnpj;
15      public $email;
16      public $id_endereco;
17
18      public function initialize(){
19          $this->belongsTo("id_endereco", "Endereco", "id");
20      }
21
22
23      public function getSchema()  {
24          return "public";
25      }
26  }
27
28  ?>
```

Código 55: Fornecedor.php

5.1.5 Funcionário

```
1   <?php
2
3   use Phalcon\Mvc\Model;
4   use Phalcon\Mvc\Model\Message;
5
6
7   class Funcionario extends Model {
8
9       public $id;
10      public $tipo;
11      public $nome;
12      public $id_endereco;
13      public $telefone;
14      public $cpf;
15      public $nome_de_usuario;
16
17      public function initialize(){
18          $this->belongsTo("id_endereco", "Endereco", "id");
19      }
20
21      public function getSchema()  {
22          return "public";
23      }
24  }
25
26  ?>
```

Código 56: Funcionario.php

5.1.6 Ingrediente

```php
<?php

use Phalcon\Mvc\Model;
use Phalcon\Mvc\Model\Message;

class Ingrediente extends Model {
    public $id_item;
    public $id_produto;
    public $quantidade;

    public function initialize(){
        $this->belongsTo("id_produto", "Produto", "id");
        $this->belongsTo("id_item", "Item", "id");
    }

    public function getSchema()  {
        return "public";
    }
}
```

Código 57: Ingrediente.php

5.1.7 Item

```php
<?php

use Phalcon\Mvc\Model;
use Phalcon\Mvc\Model\Message;


class Item extends Model {

    public $id;
    public $nome;
    public $unidade;
    public $quantidade;
    public $limite;

    public function getSchema()  {
        return "public";
    }
}

?>
```

Código 58: Item.php

5.1.8 Produto

```php
<?php

use Phalcon\Mvc\Model;
use Phalcon\Mvc\Model\Message;


class Produto extends Model {

    public $id;
    public $nome;
    public $preco;

    public function initialize(){
        $this->hasMany("id", "Ingrediente", "id_produto");
    }

    public function getSchema()  {
        return "public";
    }
}

?>
```

Código 59: Produto.php

5.2 Controles

5.2.1 Controlador Base

```php
<?php

use Phalcon\Mvc\Controller;
use Phalcon\Mvc\Dispatcher;

class BaseController extends Controller {

    public $debugArray;
    public $lastDebugKey = 1;

    /**
     * This action is available for multiple controllers
     */
    public function redis() {
        return $this->getDi()->getRedis();
    }

    // After route executed event. Convert everything to JSON response
    public function afterExecuteRoute(Dispatcher $dispatcher) {
        // If the response was sent before (404 example below)
        if ($this->response->isSent()) {
            return;
        }

        $data = $dispatcher->getReturnedValue();

        /* Set global params if is not set in controller/action */
        return $this->sendContent(200, "", $data);
    }

    public function sendContent($statusCode, $statusMessage, $content) {
        if ($this->response->isSent()) {
            return;
        }

        if ($content == null) {
            $content = array();
        } else if (!is_array($content)) {
            $content = array($content);
        }

        $content['success'] = isset($content['success']) ? $content['success'] : true;

        if ($this->debugArray) {
            $content["debug"] = $this->debugArray;
        }


        $this->response->setContentType('application/json', 'UTF-8');
        $this->response->setHeader('Access-Control-Allow-Headers', 'X-Requested-With');
        $this->response->setStatusCode($statusCode, $statusMessage);
        $this->response->setContent(json_encode($content));
        return $this->response->send();
    }


    public function notFoundAction() {
        $this->response->setStatusCode(404, 'Not Found');
        $this->response->setContentType('application/json', 'UTF-8');
        $this->response->setContent(json_encode(array("success" => false, "message" => "This endpoint doesn't exist. ".
     implode(",",$this->request->getQuery()) )));
        return $this->response->send();
    }

    public function buildErrorReturn($message) {
        return array("success" => false, "message" => $message);
    }

    public function buildReturn($data, $key = 'data') {
        return array("success" => true, $key => $data);
    }

    public function debug($obj, $key = 0) {
        if (!$this->debugArray) {
            $this->debugArray = array();
        }
        if (!$key) {
            $key = $this->lastDebugKey++;
        }
        $this->debugArray[$key] = $obj;
    }

    public function saveAction(){
        try {
            $model = ucfirst($this->router->getControllerName());

            $instance = new $model();
```

```
87              $data = json_decode($this->request->get("data"), true);
88
89              if (!$instance->save($data)){
90                  $errors = [];
91                  foreach ($instance->getMessages() as $error) {
92                      $errors[] = $error->getMessage();
93                  }
94                  return ["success" => false, "errors" => $errors];
95              } else {
96                  return ["id" => $instance->id];
97              }
98          } catch (Exception $e){
99              print_r($e);
100         }
101     }
102
103     public function findAction(){
104         try {
105             $model = ucfirst($this->router->getControllerName());
106
107             $where = $this->request->get("data");
108
109             $data = $model::find($where ? $where : null);
110
111             return ["data" => $data];
112         } catch (Exception $e){
113             print_r($e);
114         }
115     }
116
117     public function deleteAction(){
118         try {
119             $model = ucfirst($this->router->getControllerName());
120             $data = json_decode($this->request->get("data"), true);
121             if (!$data){
122                 $data = ["id" => $this->request->get("id")];
123             }
124             if ($data["id"]){
125                 $item = $model::findFirst($data["id"]);
126                 $item->delete();
127             }
128         } catch (Exception $e){
129             print_r($e);
130         }
131     }
132 }
133
134 ?>
```

Código 60: BaseController.php

5.2.2 Controlador de Endereços

```
1  <?php
2
3  class EnderecoController extends BaseController {
4
5  }
```

Código 61: EnderecoController.php

5.2.3 Controlador de Fornecedor

```
1  <?php
2
3  class FornecedorController extends BaseController {
4      public function saveAction(){
5          $fornecedor_arr = json_decode($this->request->get("data"), true);
6          $endereco_arr = $fornecedor_arr["endereco"];
7          $endereco = new Endereco();
8          $fornecedor = new Fornecedor();
9
10         if ($endereco->save($endereco_arr)){
11             $fornecedor->id_endereco = $endereco->id;
12             if (!$fornecedor->save($fornecedor_arr)){
13                 $errors = [];
14                 foreach ($fornecedor->getMessages() as $error) {
15                     $errors[] = $error->getMessage();
16                 }
17                 return ["success" => false, "errors" => $errors];
18             }
19         } else {
```

```
            //return ["success" => false, "errors" => ["nada nao"]];
            $errors = [];
            foreach ($endereco->getMessages() as $error) {
                $errors[] = $error->getMessage();
            }
            return ["success" => false, "errors" => $errors];
        }

        return ["id" => $fornecedor->id];

    }

    public function findAction(){
        try {
            $fornecedores = Fornecedor::find();
            $arr = [];
            foreach ($fornecedores as $fornecedor){
                $fornecedor->endereco;
                $fornecedor = get_object_vars($fornecedor);
                $arr[] = $fornecedor;
            }
            return ["data" => $arr];
        } catch (Exception $e){
            print_r($e);
        }
        exit;
    }
}
```

Código 62: FornecedorController.php

5.2.4 Controlador de Funcionário

```
<?php

class FuncionarioController extends BaseController {
    public function saveAction(){
        $funcionario_arr = json_decode($this->request->get("data"), true);
        $endereco_arr = $funcionario_arr["endereco"];
        $endereco = new Endereco();
        $funcionario = new Funcionario();

        if ($endereco->save($endereco_arr)){
            $funcionario->id_endereco = $endereco->id;
            if (!$funcionario->save($funcionario_arr)){
                $errors = [];
                foreach ($funcionario->getMessages() as $error) {
                    $errors[] = $error->getMessage();
                }
                return ["success" => false, "errors" => $errors];
            }
        } else {
            //return ["success" => false, "errors" => ["nada nao"]];
            $errors = [];
            foreach ($endereco->getMessages() as $error) {
                $errors[] = $error->getMessage();
            }
            return ["success" => false, "errors" => $errors];
        }

        return ["id" => $funcionario->id];

    }

    public function findAction(){
        try {
            $funcionarios = Funcionario::find();
            $arr = [];
            foreach ($funcionarios as $funcionario){
                $endereco = $funcionario->endereco;
                $funcionario = get_object_vars($funcionario);
                /*if ($endereco){
                    $funcionario["endereco"] = $endereco;
                }*/
                $arr[] = $funcionario;
            }
            return ["data" => $arr];
        } catch (Exception $e){
            print_r($e);
        }
        exit;
    }
}
```

Código 63: FuncionarioController.php

5.2.5 Controlador de Item

```php
<?php

class ItemController extends BaseController {

}
```

Código 64: ItemController.php

5.2.6 Controlador de Produto

```php


<?php

class ProdutoController extends BaseController {
    public function saveAction(){
        try {
            $produto = new Produto();
            $data = json_decode($this->request->get("data"), true);

            $produto->nome = $data["nome"];
            $produto->preco = $data["preco"];
            $ingredientesArr = $data["ingredientes"];

            if ($produto->save($data)){
                foreach ($ingredientesArr as $ingredienteArr) {
                    $item = $ingredienteArr["item"];
                    $ingrediente = new Ingrediente();
                    $ingrediente->id_item = $item["id"];
                    $ingrediente->id_produto = $produto->id;
                    $ingrediente->quantidade = $ingredienteArr["quantidade"];
                    if (!$ingrediente->save()){
                        $errors = [];
                        foreach ($ingrediente->getMessages() as $error) {
                            $errors[] = $error->getMessage();
                        }
                        return ["success" => false, "errors" => $errors];
                    }
                }
                return ["id" => $produto->id];
            } else {
                $errors = [];
                foreach ($produto->getMessages() as $error) {
                    $errors[] = $error->getMessage();
                }
                return ["success" => false, "errors" => $errors];
            }
        } catch (Exception $e){
            print_r($e);
        }
    }

    public function findAction(){
        try {
            $produtos = Produto::find();
            $arr = [];
            foreach ($produtos as $produto){
                $ingredientes = $produto->getIngrediente();
                $ingredientes_arr = $ingredientes->toArray();
                for ($i = 0; $i < count($ingredientes_arr); $i++){
                    $ingredientes_arr[$i]["item"] = $ingredientes[$i]->getItem();
                }
                $produto = get_object_vars($produto);
                if ($ingredientes){
                    $produto["ingredientes"] = $ingredientes_arr;
                }
                $arr[] = $produto;
            }
            return ["data" => $arr];
        } catch (Exception $e){
            print_r($e);
        }
        exit;
    }
}
```

Código 65: ProdutoController.php

5.3 Banco de Dados

```sql
DROP TABLE endereco CASCADE;
DROP TABLE funcionario CASCADE;
DROP TABLE mesa CASCADE;
DROP TABLE comanda CASCADE;
DROP TABLE produto CASCADE;
DROP TABLE pedido CASCADE;
DROP TABLE item CASCADE;
DROP TABLE ingrediente CASCADE;
DROP TABLE fornecedor CASCADE;
DROP TABLE compra CASCADE;

CREATE TABLE endereco (
    id serial,
    logradouro varchar(128),
    rua varchar(255),
    numero integer,
    bairro varchar(128),
    cidade varchar(128),
    estado varchar(128),
    CEP varchar(32),
    CONSTRAINT pk_endereco PRIMARY KEY (id)
);

CREATE TABLE funcionario (
    id serial,
    tipo integer,
    cpf varchar(20),
    nome varchar(128),
    telefone varchar(20),
    id_endereco integer,
    nome_usuario varchar(64),
    CONSTRAINT pk_funcionario PRIMARY KEY (id),
    CONSTRAINT fk_id_endereco FOREIGN KEY (id_endereco) REFERENCES endereco (id),
    CONSTRAINT unique_cpf UNIQUE (cpf)
);

CREATE TABLE mesa (
    numero integer,
    CONSTRAINT pk_mesa PRIMARY KEY (numero)
);

--uma comanda esta sempre associada a uma mesa
CREATE TABLE comanda (
    id serial,
    data timestamp,
    numero_mesa integer,
    CONSTRAINT pk_comanda PRIMARY KEY (id),
    CONSTRAINT fk_comanda_mesa FOREIGN KEY (numero_mesa) REFERENCES mesa (numero)
);

CREATE TABLE produto (
    id serial,
    nome varchar(128),
    preco float,
    CONSTRAINT pk_produto PRIMARY KEY (id)
);

--um pedido pode estar em apenas uma comanda, e possui apenas um produto.
CREATE TABLE pedido (
    id serial,
    quantidade integer,
    entrege boolean,
    id_produto integer,
    id_comanda integer,
    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,
    unidade varchar(3),
    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 66: CriarBanco.sql

6 Diagramas

6.1 Diagrama de Classes

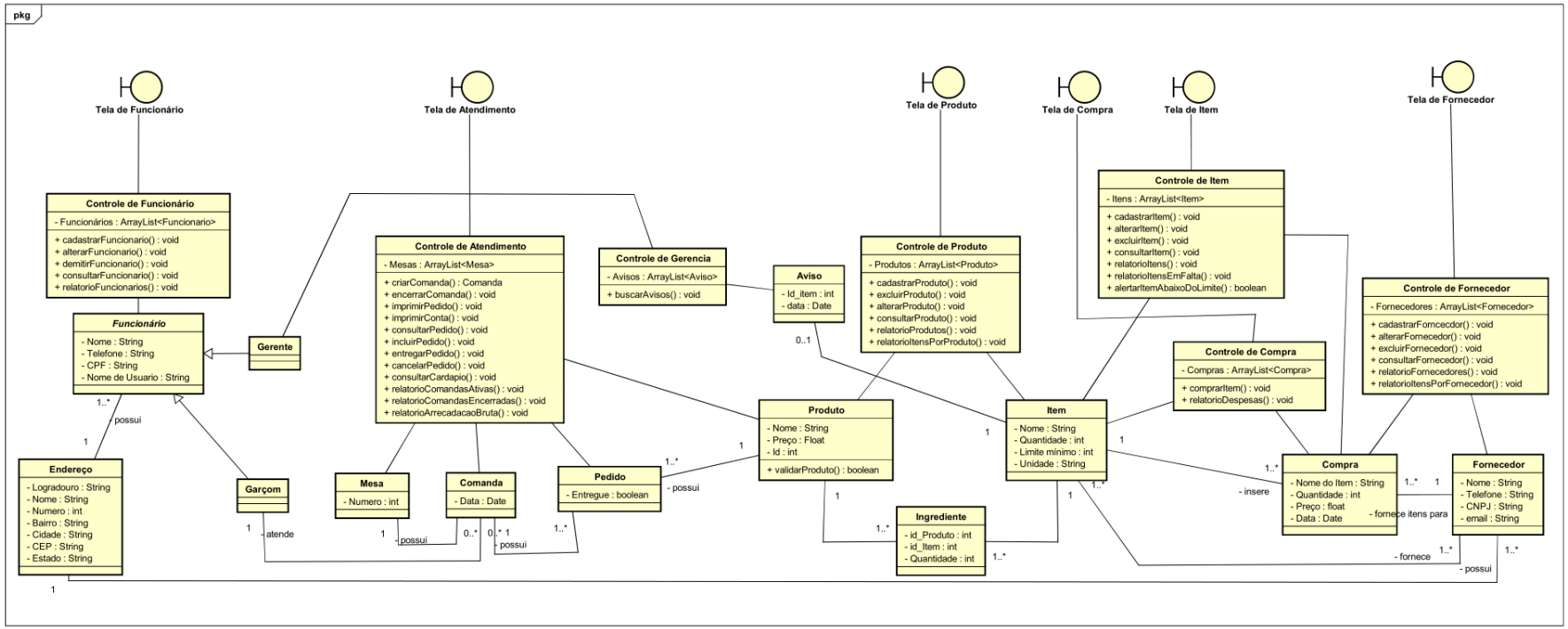

Figura 1: Diagrama de Classes

6.2 Diagrama de Sequência

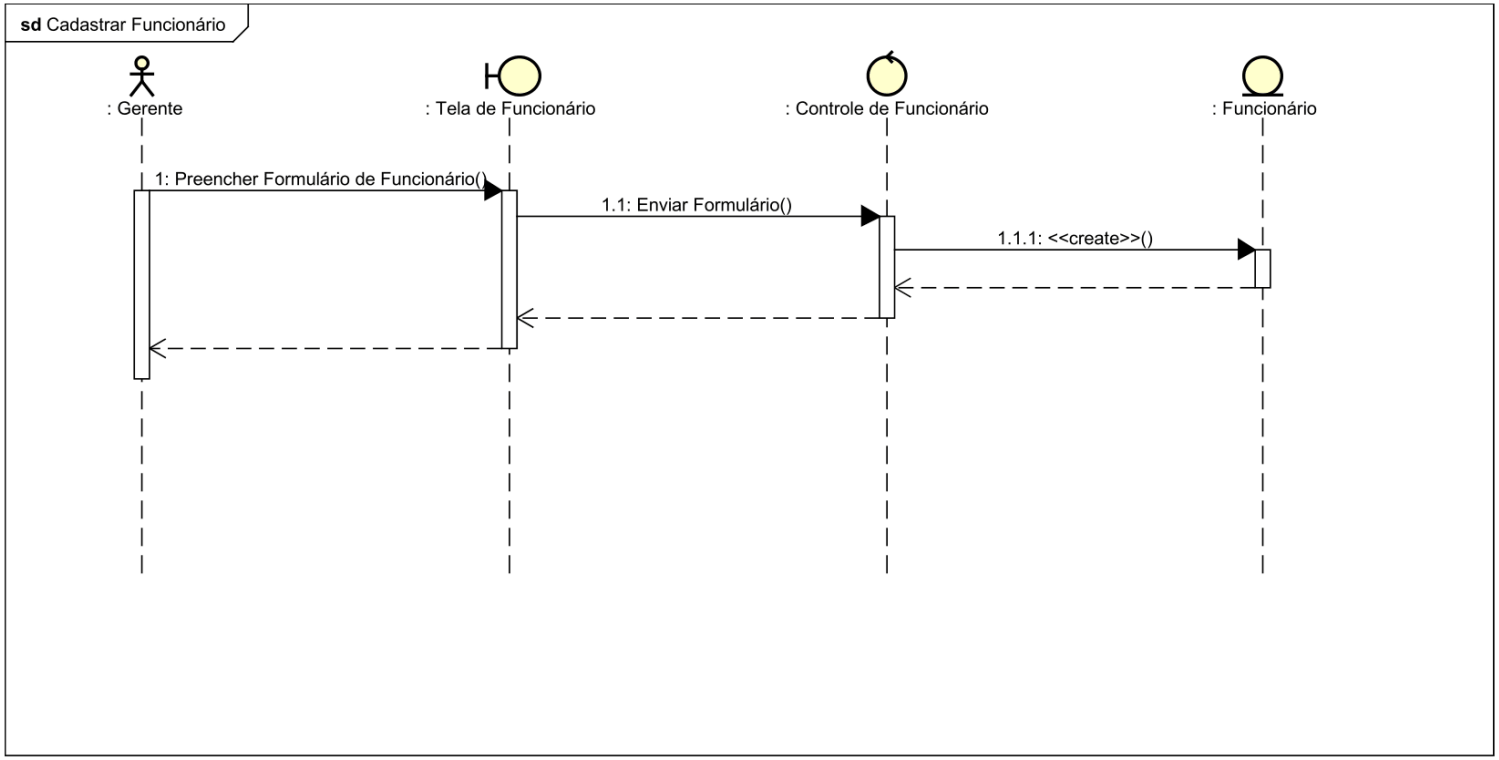

Figura 2: Cadastrar Funcionário

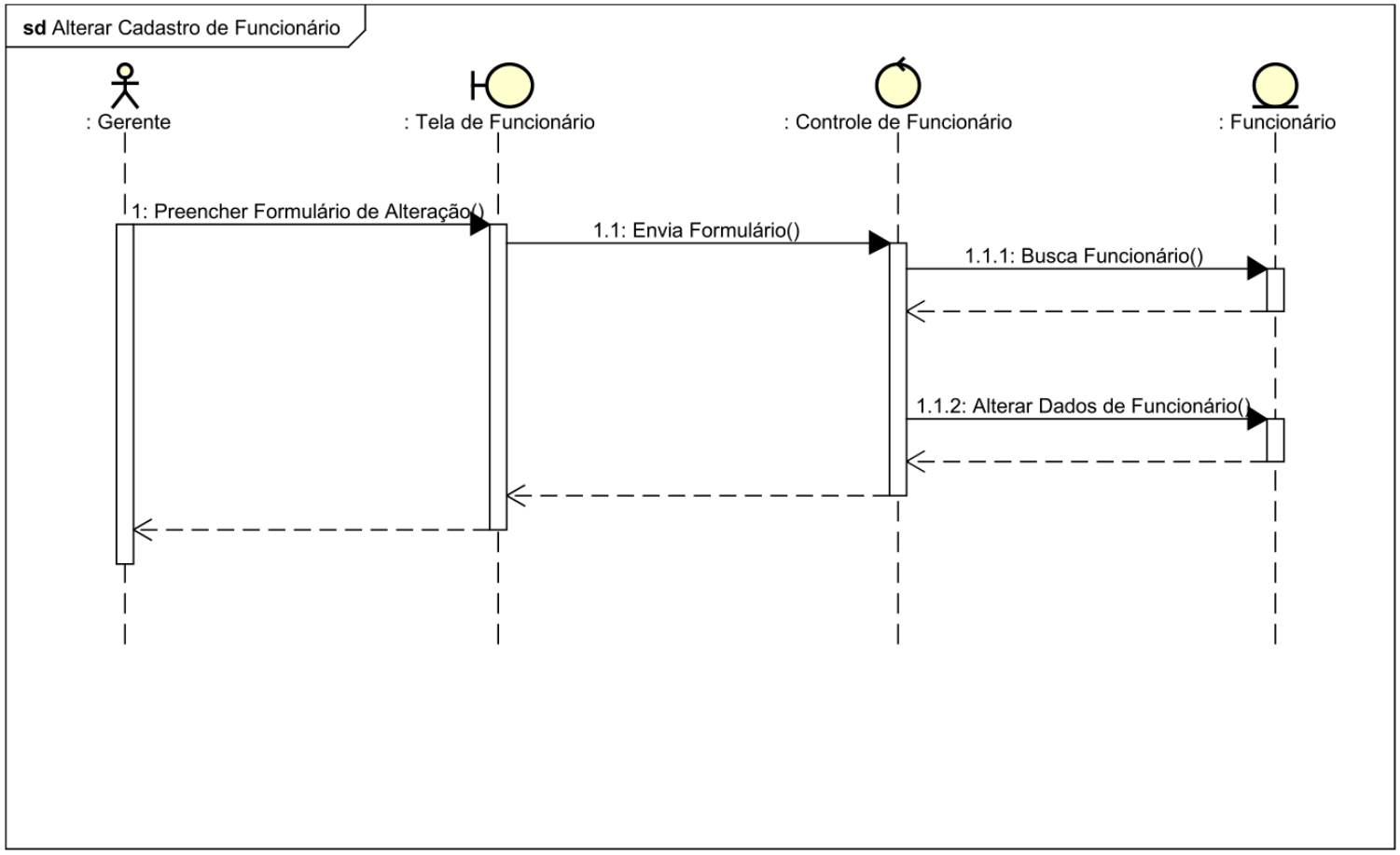

Figura 3: Alterar Cadastro de Funcionário

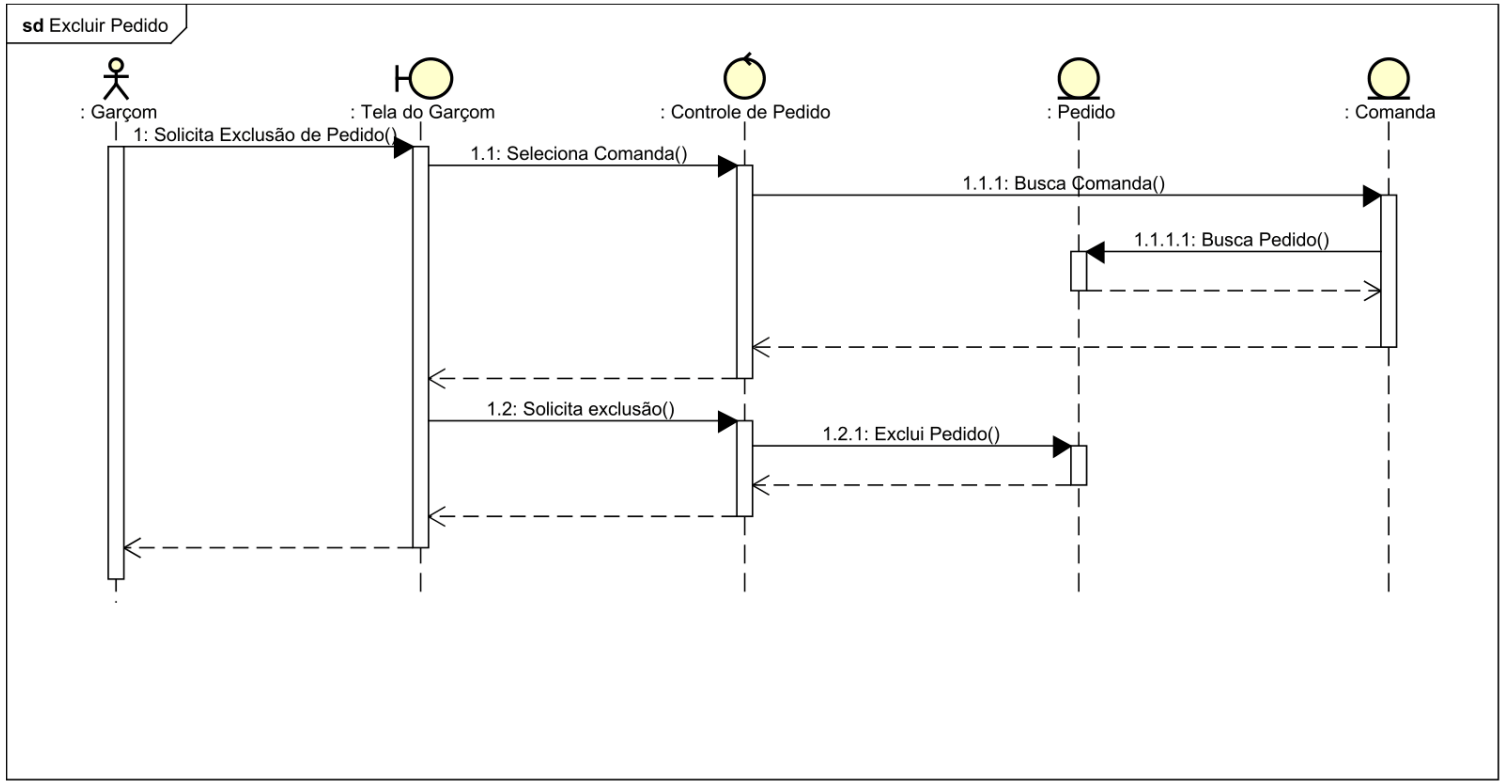

Figura 4: Excluir Pedido

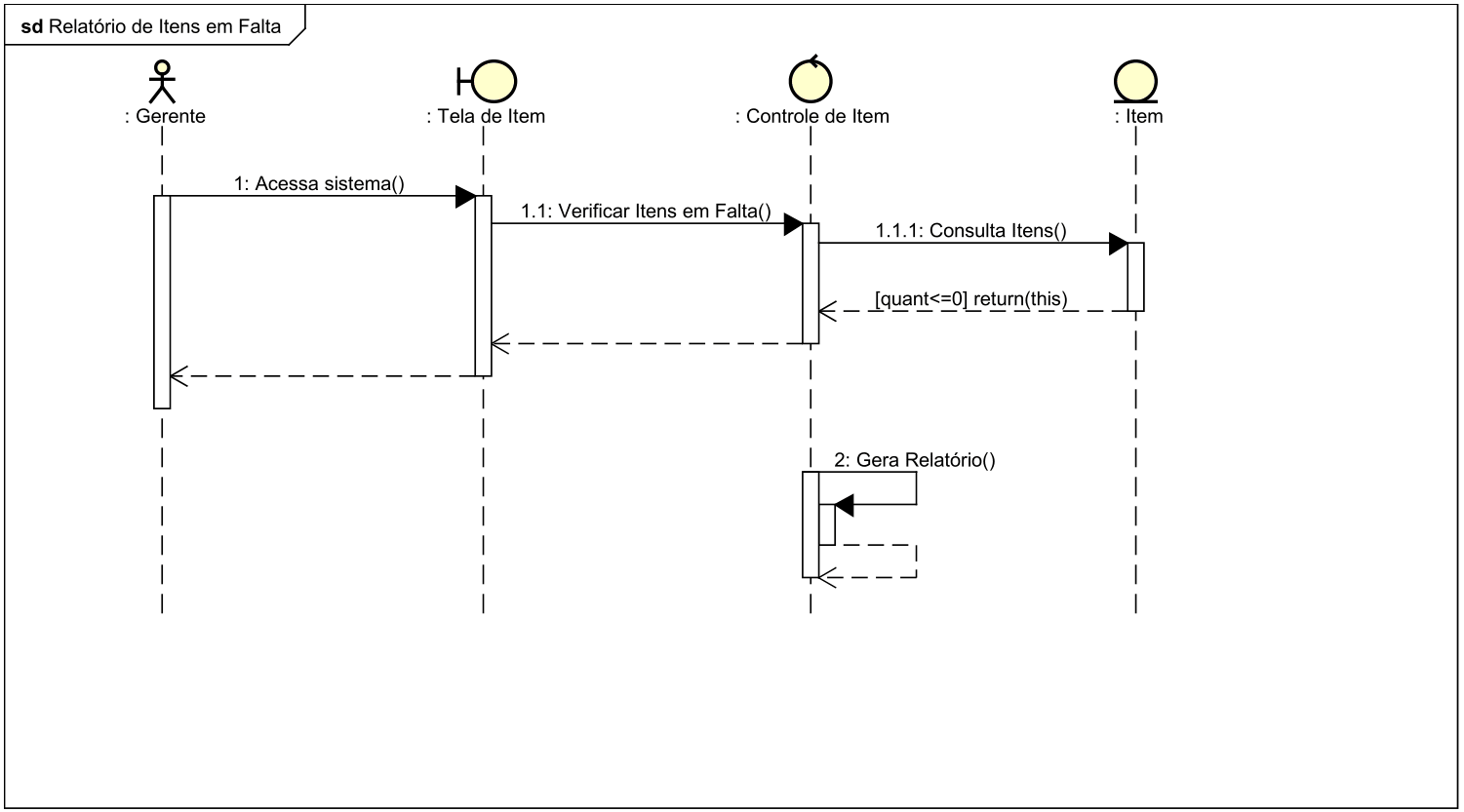

Figura 5: Relatório de Itens em Falta

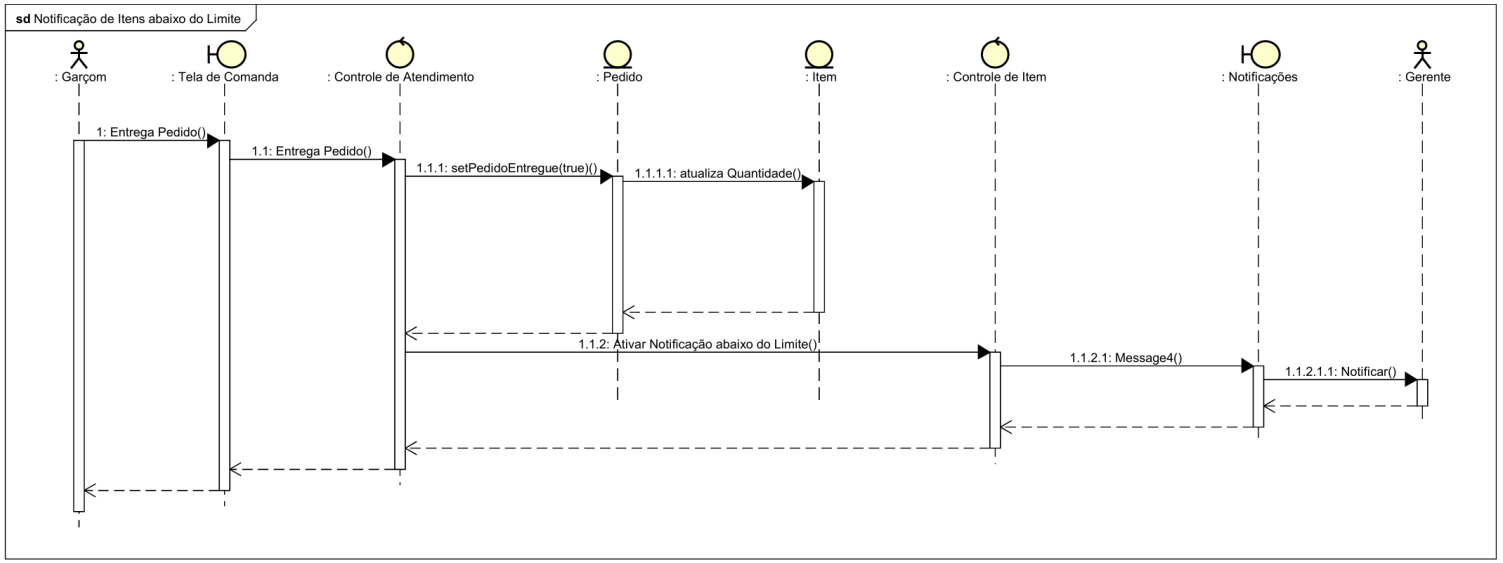

Figura 6: Notificação de Itens abaixo do limite

6.3 Diagrama de Casos de Uso

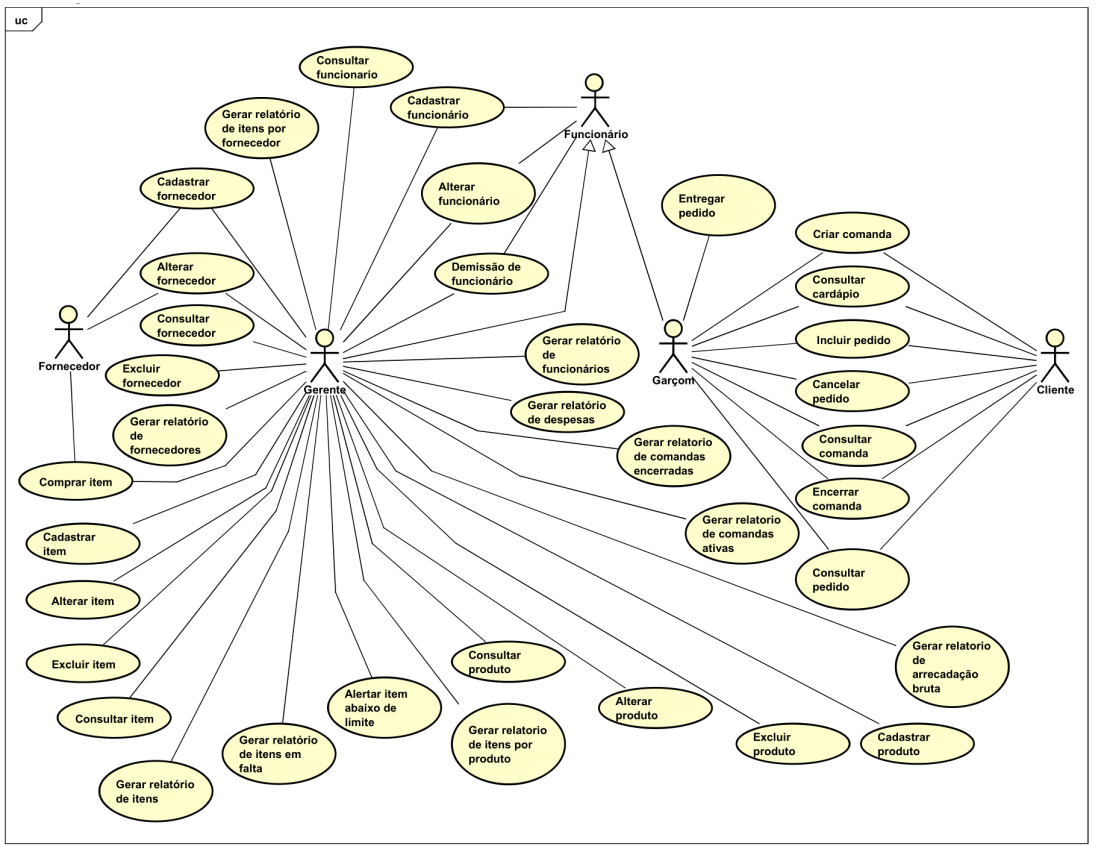

Figura 7: Diagrama de Casos de Uso