



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

Documentos Desenvolvimento de Software

Prof. Dr. MICHEL DOS SANTOS SOARES

São Cristóvão, Sergipe
Maio – 2016

Componentes:

ELTON MOREIRA CARVALHO – 201310004602

FERNNADO MELO NASCIMENTO – 201210009310

FERNNADO MESSIAS DOS SANTOS – 201320001408

RODRIGO BENEDITO OTONI – 201210009188

THALES FRANCISCO SOUZA SAMPAIO ALVES DOS SANTOS – 201210012648

Conteúdo

| | | |
|----------|--|-----------|
| 1 | Levantamento de Requisitos | 6 |
| 1.1 | Propósito do Documento | 6 |
| 1.2 | Escopo do Produto | 6 |
| 1.3 | Definições e Abreviações | 6 |
| 1.3.1 | Definições | 6 |
| 1.3.2 | Abreviações | 6 |
| 1.4 | Referências | 6 |
| 1.5 | Visão Geral do Restante do Documento | 7 |
| 1.6 | Descrição Geral | 7 |
| 1.6.1 | Perspectiva do Produto | 7 |
| 1.6.2 | Funções do Produto | 7 |
| 1.6.3 | Características do Usuário | 7 |
| 1.6.4 | Restrições Gerais | 7 |
| 1.6.5 | Suposições e Dependências | 7 |
| 1.7 | Requisitos específicos | 8 |
| 1.7.1 | Requisitos Funcionais | 8 |
| 1.7.2 | Requisitos Não Funcionais | 10 |
| 2 | Plano de Projeto | 12 |
| 2.1 | Motivação | 12 |
| 3 | Casos de Uso | 13 |
| 4 | Codificação referente ao Cliente (Java/Android) | 16 |
| 4.1 | Modelos | 16 |
| 4.1.1 | Model | 16 |
| 4.2 | Aviso | 17 |
| 4.2.1 | Comanda | 17 |
| 4.2.2 | Compra | 18 |
| 4.2.3 | Endereço | 18 |
| 4.2.4 | Fornecedor | 19 |
| 4.2.5 | Funcionário | 20 |
| 4.2.6 | Garçom | 21 |
| 4.2.7 | Gerente | 21 |
| 4.2.8 | Ingrediente | 21 |
| 4.2.9 | Item | 22 |
| 4.2.10 | Mesa | 22 |
| 4.2.11 | Pedido | 23 |
| 4.2.12 | Produto | 23 |
| 4.3 | Controles | 24 |
| 4.3.1 | Controle de Atendimento | 24 |
| 4.3.2 | Controle de Compra | 25 |
| 4.3.3 | Controle de Fornecedor | 26 |
| 4.3.4 | Controle de Funcionário | 28 |

| | | |
|----------|--|-----------|
| 4.3.5 | Controle de Gerência | 29 |
| 4.3.6 | Controle de Impressão | 29 |
| 4.3.7 | Controle de Item | 30 |
| 4.3.8 | Controle de Produto | 31 |
| 4.4 | Fronteiras | 32 |
| 4.4.1 | Main Activity | 32 |
| 4.4.2 | Menu Principal | 32 |
| 4.4.3 | Menu Fornecedores | 33 |
| 4.4.4 | Menu Funcionários | 34 |
| 4.4.5 | Menu Garçom | 34 |
| 4.4.6 | Menu Itens | 35 |
| 4.4.7 | Menu Produtos | 35 |
| 4.4.8 | Menu Relatório | 36 |
| 4.4.9 | Busca Fornecedor | 36 |
| 4.4.10 | Busca Funcionário | 38 |
| 4.4.11 | Busca Item | 39 |
| 4.4.12 | Busca Produto | 40 |
| 4.4.13 | Cadastro Endereço | 40 |
| 4.4.14 | Cadastro Fornecedor | 41 |
| 4.4.15 | Cadastro Funcionário | 43 |
| 4.4.16 | Cadastro Item | 44 |
| 4.4.17 | Cadastro Produto | 45 |
| 4.4.18 | Criação Comanda | 47 |
| 4.4.19 | Exibir Pedido | 47 |
| 4.4.20 | Selecionar Ingredientes | 48 |
| 4.4.21 | Tela Garçom | 48 |
| 4.4.22 | Tela Relatórios | 48 |
| 4.5 | Adapters | 49 |
| 4.5.1 | Fornecedor Adapter | 49 |
| 4.5.2 | Funcionario Adapter | 49 |
| 4.5.3 | Item Adapter | 50 |
| 4.5.4 | Produto Adapter | 51 |
| 4.6 | Helpers | 52 |
| 4.6.1 | Server Request | 52 |
| 4.6.2 | Request Callback | 53 |
| 4.6.3 | Utilitários | 53 |
| 5 | Codificação referente ao Servidor (PHP) | 56 |
| 5.1 | Modelos | 56 |
| 5.1.1 | Comanda | 56 |
| 5.1.2 | Compra | 56 |
| 5.1.3 | Endereço | 56 |
| 5.1.4 | Fornecedor | 57 |
| 5.1.5 | Funcionário | 57 |
| 5.1.6 | Ingrediente | 58 |
| 5.1.7 | Item | 58 |

| | | |
|----------|--------------------------------------|-----------|
| 5.1.8 | Produto | 58 |
| 5.2 | Controles | 59 |
| 5.2.1 | Controlador Base | 59 |
| 5.2.2 | Controlador de Endereços | 60 |
| 5.2.3 | Controlador de Fornecedor | 60 |
| 5.2.4 | Controlador de Funcionário | 61 |
| 5.2.5 | Controlador de Item | 62 |
| 5.2.6 | Controlador de Produto | 62 |
| 5.3 | Banco de Dados | 62 |
| 6 | Diagramas | 65 |
| 6.1 | Diagrama de Classes | 65 |
| 6.2 | Diagrama de Sequência | 65 |
| 6.3 | Diagrama de Casos de Uso | 71 |

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 líquida do restaurante em um intervalo de tempo, em ordem de dias, definido pelo gerente.

1.7.2 Requisitos Não Funcionais

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

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

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

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

2 Plano de Projeto

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 .

O prazo necessário para conclusão das etapas de desenvolvimento e implantação do projeto foi estimado em 132 dias úteis.

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

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



```

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

```

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
15     public Comanda(Context context, String nome, Pedido primeiro){
16         super(context);
17         Date agora = new Date();
18         pedidos.add(primeiro);
19         data = agora.toString();
20         this.nome = nome;
21         ativa = true;
22     }
23
24     public void addPedido(Pedido outro){
25         pedidos.add(outro);
26     }
27
28     public void setNome(String nome){
29         this.nome=nome;
30     }
31
32     public ArrayList< Pedido > getPedidos(){
33         return pedidos;
34     }
35
36     public Pedido getPedido(int indice){

```

```

37     return pedidos.get(indice);
38 }
39
40 public void removerPedido(Pedido pedido){
41     pedidos.remove(pedido);
42 }
43
44 public boolean estaAtiva(){
45     return ativa;
46 }
47
48 public void desativar(){
49     ativa = false;
50 }
51
52 public float getCustoTotal(){
53     float custo = 0;
54     for (Pedido pedido : pedidos){
55         custo+= pedido.getCusto();
56     };
57     return custo;
58 }
59 }

```

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; }
42 }

```

Código 4: Compra.java

4.2.3 Endereço

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 public class Endereco extends Model<Endereco> {
6     private String logradouro;
7     private String rua;

```

```

8     private int numero;
9     private String bairro;
10    private String cidade;
11    private String estado;
12    private String cep;
13
14    public Endereco(Context context, String logradouro, String rua, int numero, String bairro, String cidade, String
15        estado, String cep) {
16        super(context);
17        this.logradouro = logradouro;
18        this.rua = rua;
19        this.numero = numero;
20        this.bairro = bairro;
21        this.cidade = cidade;
22        this.cep = cep;
23        this.estado = estado;
24    }
25
26    public String getLogradouro() {
27        return logradouro;
28    }
29
30    public String getRua() {
31        return rua;
32    }
33
34    public int getNumero() {
35        return numero;
36    }
37
38    public String getBairro() {
39        return bairro;
40    }
41
42    public String getCidade() {
43        return cidade;
44    }
45
46    public String getCep() {
47        return cep;
48    }
49
50    public String getEstado() {
51        return estado;
52    }
53
54    public void setEstado(String estado) {
55        this.estado = estado;
56    }
57 }

```

Código 5: Endereco.jav

4.2.4 Fornecedor

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 public class Fornecedor extends Model<Fornecedor> {
6     private String nome;
7     private String telefone;
8     private String cnpj;
9     private String email;
10    private Endereco endereco;
11
12    public Fornecedor(Context context){
13        super(context);
14    }
15
16    public Fornecedor(Context context, String nome, String telefone, String cnpj, String email){
17        super(context);
18        this.nome = nome;
19        this.telefone = telefone;
20        this.cnpj = cnpj;
21        this.email = email;
22    }
23
24    public String getNome(){
25        return nome;
26    }
27
28    public String getTelefone (){
29        return telefone;
30    }
31
32    public String getCnpj(){
33        return cnpj;
34    }
35 }

```

```

34     }
35
36     public String getEmail(){
37         return email;
38     }
39
40     public void setNome(String nome){
41         this.nome = nome;
42     }
43
44     public void setTelefone (String telefone){
45         this.telefone = telefone;
46     }
47
48     public void setCnpj(String cnpj){
49         this.cnpj = cnpj;
50     }
51
52     public void setEmail(String email){
53         this.email = email;
54     }
55
56     public Endereco getEndereco() {
57         return endereco;
58     }
59
60     public void setEndereco(Endereco endereco) {
61         this.endereco = endereco;
62     }
63 }

```

Código 6: Fornecedor.java

4.2.5 Funcionário

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4 import android.content.Intent;
5
6 public class Funcionario extends Model<Funcionario> {
7     private String nome;
8     private Integer id_endereco;
9     private String telefone;
10    private String cpf;
11    private String nome_de_usuario;
12    private Integer tipo;
13
14    public Funcionario(Context context){
15        super(context);
16    }
17
18    public Funcionario(Context context, String nome, Integer id_endereco, String telefone, String cpf, String
19        nome_de_usuario, Integer tipo) {
20        super(context);
21        this.nome = nome;
22        this.id_endereco = id_endereco;
23        this.telefone = telefone;
24        this.cpf = cpf;
25        this.nome_de_usuario = nome_de_usuario;
26        this.tipo = tipo;
27    }
28
29    public String getNome() {
30        return nome;
31    }
32
33    public Integer getIdEndereco() {
34        return id_endereco;
35    }
36
37    public String getTelefone() {
38        return telefone;
39    }
40
41    public String getCpf() {
42        return cpf;
43    }
44
45    public String getNome_de_usuario() {
46        return nome_de_usuario;
47    }
48
49    public Integer getTipo () { return tipo; };
50
51    public void setNome(String nome){
52        this.nome = nome;
53    }
54
55 }

```

```

53
54     public void setTelefone (String telefone){
55         this.telefone = telefone;
56     }
57
58     public void setCpf(String cnpj){
59         this.cpf = cnpj;
60     }
61
62     public void setIdEndereco(Integer id_endereco) {
63         this.id_endereco = id_endereco;
64     }
65
66     public void setNome_de_usuario (String nome_de_usuario) { this.nome_de_usuario = nome_de_usuario; }
67
68     public void setTipo (Integer tipo) {this.tipo = tipo; };
69 }

```

Código 7: Funcionario.java

4.2.6 Garcom

```

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, Integer id_endereco, String telefone, String cpf, String
7         nome_de_usuario, Integer tipo) {
8         super(context, nome, id_endereco, telefone, cpf, nome_de_usuario, tipo);
9     }
10 }

```

Código 8: Garcom.java

4.2.7 Gerente

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 public class Gerente extends Funcionario {
6     public Gerente(Context context, String nome, Integer id_endereco, String telefone, String cpf, String
7         nome_de_usuario, Integer tipo) {
8         super(context, nome, id_endereco, telefone, cpf, nome_de_usuario, tipo);
9     }
10 }

```

Código 9: Gerente.java

4.2.8 Ingrediente

```

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

```

Código 10: Ingrediente.jav

4.2.9 Item

```
1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 import ds2.equipe1.restaurante.controles.ControleDeAtendimento;
6
7 public class Item extends Model<Item> {
8     private String nome;
9     private String unidade;
10    private int quantidade;
11    private int limiteMinimo;
12
13
14    public Item(Context context){
15        super(context);
16    }
17
18    public Item(Context context, String nome, int quantidade, String unidade, int limiteMinimo){
19        super(context);
20        this.nome = nome;
21        this.unidade = unidade;
22        this.quantidade = quantidade;
23        this.limiteMinimo = limiteMinimo;
24    }
25
26    public void verificarItemAbaixoDoLimite(int quantidadeParaReduzir){
27        if (getQuantidade()-quantidadeParaReduzir < getLimiteMinimo()){
28            new Aviso(context, this).save();
29        }
30    }
31
32    public String getNome(){
33        return nome;
34    }
35
36    public int getQuantidade(){
37        return quantidade;
38    }
39
40    public int getLimiteMinimo(){
41        return limiteMinimo;
42    }
43
44    public void setQuantidade(int quantidade){
45        //alterar quantidade no banco
46        this.quantidade = quantidade;
47    }
48
49    public void setLimiteMinimo(int limiteMinimo){
50        //alterar limiteMinimo no banco
51        this.limiteMinimo = limiteMinimo;
52    }
53
54    public void setNome(String nome){
55        this.nome = nome;
56    }
57
58    public void setUnidade(String unidade){
59        this.unidade=unidade;
60    }
61 }
```

Código 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 Comanda getComanda(int indice){
37         return comandas.get(indice);
38     }
39 }

```

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;
11
12 public Produto(Context context, String nome, float preco, ArrayList<Ingrediente> ingredientes) {
13     super(context);
14     this.nome = nome;
15     this.preco = preco;
16     this.ingredientes = ingredientes;
17 }
18
19 /*public static ArrayList<Produto> carregarProdutos() {
20     ArrayList<Produto> listaDeProdutos = new ArrayList<Produto>();
21
22     ArrayList<Produto> lista = new ArrayList<Produto>();
23
24     //conexao com o banco estabelecida
25     for (int i = 0; i < lista.count(); i++) {
26         listaDeProdutos.add(i, new Produto(lista.nome, lista.preco, lista.id, ...));
27     }
28
29     return listaDeProdutos; //(???)
30 }*/
31
32 public boolean validarProduto() {
33     /*for (int i = 0; i < ingredientes.size(); i++) {
34         if (ingredientes.get(i).getQuantidade() > ControleDeItem.consultarItem(ingredientes.get(i).getItem().
35             getNome(), new RequestCallback<Item>() {
36                 @Override
37                 public void execute(ArrayList<Item> itens) {
38                     BuscaItem.this.itens.clear();
39                     BuscaItem.this.itens.addAll(itens);
40                     adapter.notifyDataSetChanged();
41                     super.execute(itens);
42                 }
43             }).getQuantidade()) {
44                 return false;
45             }
46     }*/
47     return true;
48 }
49
50 //Deve ser chamado quando o garcom conclui o pedido.
51 public void alertarSobreItensAbaixoDoLimite(){
52     for (Ingrediente ingrediente : getIngredientes()){
53         Item item = ingrediente.getItem();
54         item.verificarItemAbaixoDoLimite(ingrediente.getQuantidade());
55     }
56 }
57
58 public String getNome(){
59     return nome;
60 }
61
62 public float getPreco(){
63     return preco;
64 }
65
66 public ArrayList <Ingrediente> getIngredientes(){
67     return ingredientes;
68 }
69
70 public void setPreco(float preco){
71     this.preco = preco;
72 }
73 }

```

Código 14: Produto.java

4.3 Controles

4.3.1 Controle de Atendimento

```

1 package ds2.equipe1.restaurante.controles;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 import ds2.equipe1.restaurante.modelos.Comanda;
8 import ds2.equipe1.restaurante.modelos.Mesa;
9 import ds2.equipe1.restaurante.modelos.Pedido;
10 import ds2.equipe1.restaurante.modelos.Produto;
11
12 public class ControleDeAtendimento {
13     private Context context;
14     private ControleDeImpressao controleDeImpressao;
15 }

```



```

16 private ArrayList<Mesa> mesas = new ArrayList<Mesa>();
17
18 public ControleDeAtendimento(Context context){
19     this.context = context;
20     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, 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 + "\n10% = " + 0.1*total + "\nTotal" + 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                 }, null);
50         } else {
51             try {
52                 ArrayList<Compra> comprasFiltradas = new ArrayList<>();
53
54                 for (Compra compra : compras) {
55                     if (compra.getNomeDoItem().contains(consulta)) {
56                         comprasFiltradas.add(compra);
57                     }
58                 }
59
60                 if (callback != null) {
61                     callback.execute(comprasFiltradas);
62                 }
63             } catch (Exception e) {
64                 Log.e(Utils.TAG, "Erro ao consultar fornecedores:", e);
65             }
66         }
67     }
68
69     public static void selecionarParaEditar(Compra compra){
70         ControleDeCompra.selecionada = compra;
71     }
72
73     public static Compra getSelecionada(){
74         return selecionada;
75     }
76
77     public static void deselecionar(){
78         selecionada = null;
79     }
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 import android.widget.Toast;
6
7 import com.androidquery.callback.AjaxCallback;
8 import com.google.gson.reflect.TypeToken;
9
10 import java.lang.reflect.Array;
11 import java.util.ArrayList;
12
13 import ds2.equipe1.restaurante.helpers.RequestCallback;
14 import ds2.equipe1.restaurante.helpers.Utils;
15 import ds2.equipe1.restaurante.modelos.Fornecedor;
16 import ds2.equipe1.restaurante.modelos.Item;
17 import ds2.equipe1.restaurante.modelos.Model;
18
19 public class ControleDeFornecedor {
20     //Lista com a consulta mais recente de fornecedores no servidor.
21     private ArrayList<Fornecedor> fornecedores = new ArrayList<>();
22     //Essa variavel serve para deixar uma referencia na memoria o tempo inteiro de qual esta selecionado,
23     //para que quando haja uma edicao, a alteracao seja refletida em mais de uma tela (exemplo: editar um fornecedor e
24     //atualizar na tela de busca).
25     private static Fornecedor selecionado;
26     private Context context;
27
28     public ControleDeFornecedor(Context context){
29         this.context = context;
30     }
31
32     public void salvarFornecedor(Fornecedor fornecedor){
33         fornecedor.save();
34     }
35
36     public boolean excluirFornecedor(Fornecedor fornecedor){
37         if (fornecedor.getId() != -1) {
38             fornecedor.delete();
39             return true;
40         } else return false;
41     }
42
43     public void consultarFornecedor(String consulta, final RequestCallback<Fornecedor> callback){
44         //Se a consulta for vazio, pega todos os itens do banco de dados, e coloca na memoria ram
45         if (consulta.isEmpty()) {
46             Model.find(context, Fornecedor.class, new TypeToken<ArrayList<Fornecedor>>() {
47                 @Override
48                 public void execute(ArrayList<Fornecedor> lista) throws Exception {
49                     super.execute(lista);
50
51                     fornecedores.clear();
52                     fornecedores.addAll(lista);
53
54                     if (callback != null){
55                         callback.execute(lista);
56                     }
57                 }
58             }, null);
59         } else {
60             //Se tiver consulta, faz a pesquisa nos itens que ja estao na memoria ram
61             try {
62                 ArrayList<Fornecedor> fornecedoresFiltrados = new ArrayList<>();
63
64                 for (Fornecedor fornecedor : fornecedores) {
65                     if (fornecedor.getNome().contains(consulta) || fornecedor.getCnpj().contains(consulta)) {
66                         fornecedoresFiltrados.add(fornecedor);
67                     }
68                 }
69
70                 if (callback != null) {
71                     callback.execute(fornecedoresFiltrados);
72                 }
73             } catch (Exception e){
74                 Log.e(Utils.TAG, "Erro ao consultar fornecedores: ", e);
75             }
76         }
77     }
78
79     public void relatorioFornecedor(Fornecedor fornecedor){
80         //Banco de dados
81         //Exibir Relatorio
82     }
83
84     public void relatorioItensPorFornecedor(Item item, Fornecedor fornecedor){
85         //Banco de Dados
86         //Exibir Relatorio
87     }
88
89     public static void selecionarParaEditar(Fornecedor fornecedor){
90         ControleDeFornecedor.selecionado = fornecedor;
91     }
92
93     public static Fornecedor getSelecionado(){
94         return selecionado;
95     }

```

```

95     }
96
97     public static void deselecionar(){
98         selecionado = null;
99     }
100 }

```

Código 17: ControleDeFornecedor.java

4.3.4 Controle de Funcionário

```

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.Funcionario;
13 import ds2.equipe1.restaurante.modelos.Model;
14
15 public class ControleDeFuncionario {
16     private ArrayList<Funcionario> funcionarios = new ArrayList<>();
17
18     private static Funcionario selecionado;
19     private Context context;
20
21     private ControleDeImpressao controleDeImpressao;
22
23     public ControleDeFuncionario(Context context){
24         controleDeImpressao = new ControleDeImpressao(context);
25     }
26
27     public void salvarFuncionario(Funcionario funcionario){
28         funcionario.save();
29     }
30     public void cadastrarFuncionario(Funcionario funcionario){
31         funcionarios.add(funcionario);
32         funcionario.save();
33
34         //Banco de Dados
35         //SQL (Sem o Endereco)
36         //INSERT INTO funcionario (nome, telefone, cnpj, email) VALUES (funcionario.getRua(), funcionario.getTelefone()
37         , funcionario.getCnpj(), funcionario.getEmail());
38     }
39
40     public boolean alterarFuncionario(Funcionario funcionario){
41         if (funcionario.getId() != -1) {
42             funcionario.save();
43             return true;
44         } else return false;
45
46         //Banco de Dados
47         //SQL (Sem o Endereco)
48         //UPDATE funcionario SET nome = funcionario.getRua(), telefone = funcionario.getRua(), email = funcionario.
49         getEmail WHERE cnpj = funcionario.getCnpj();
50     }
51
52     public boolean excluirFuncionario(Funcionario funcionario){
53         if (funcionario.getId() != -1) {
54             funcionario.delete();
55             return true;
56         } else return false;
57     }
58
59     public void consultarFuncionario(String consulta, final RequestCallback<Funcionario> callback){
60         //Se a consulta for vazia, pega todos os itens do banco de dados e coloca na memoria ram
61         if (consulta.isEmpty()) {
62             Model.find(context, Funcionario.class, new TypeToken<ArrayList<Funcionario>>() {
63             }.getType(), new RequestCallback<Funcionario>() {
64             @Override
65             public void execute(ArrayList<Funcionario> lista) throws Exception {
66                 super.execute(lista);
67
68                 funcionarios.clear();
69                 funcionarios.addAll(lista);
70
71                 if (callback != null){
72                     callback.execute(lista);
73                 }
74             }
75             }, null);
76         } else {
77             //Se tiver consulta, faz a pesquisa nos itens que ja estao na memoria ram

```

```

76         try {
77             ArrayList<Funcionario> funcionariosFiltrados = new ArrayList<>();
78
79             for (Funcionario funcionario : funcionarios) {
80                 if (funcionario.getNome().contains(consulta) || funcionario.getCpf().contains(consulta)) {
81                     funcionariosFiltrados.add(funcionario);
82                 }
83             }
84
85             if (callback != null) {
86                 callback.execute(funcionariosFiltrados);
87             }
88         } catch (Exception e){
89             Log.e(Utils.TAG, "Erro ao consultar funcionarios: ", e);
90         }
91     }
92 }
93
94 public void relatorioFuncionario(Funcionario funcionario){
95     //Banco de dados
96     //Exibir Relatorio
97 }
98
99 public static void selecionarParaEditar(Funcionario funcionario){
100     ControleDeFuncionario.selecionado = funcionario;
101 }
102
103 public static Funcionario getSelecionado(){
104     return selecionado;
105 }
106
107 public static void deselecionar(){
108     selecionado = null;
109 }
110
111 }

```

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

```
1 package ds2.equipe1.restaurante.controles;
2
3 import android.content.Context;
4 import android.util.Log;
5
6 import com.google.gson.reflect.TypeToken;
7
8 import java.util.ArrayList;
9
10 import ds2.equipe1.restaurante.helpers.RequestCallback;
11 import ds2.equipe1.restaurante.helpers.Utils;
12 import ds2.equipe1.restaurante.modelos.Fornecedor;
13 import ds2.equipe1.restaurante.modelos.Item;
14 import ds2.equipe1.restaurante.modelos.Model;
15
16 public class ControleDeItem {
17     //Lista com a consulta mais recente de itens no servidor.
18     private ArrayList<Item> itens = new ArrayList<>();
19     private Context context;
20     private static Item selecionado;
21
22     public ControleDeItem(Context context) {
23         this.context = context;
24     }
25
26     public static void salvarItem(Item item) {
27         item.save();
28     }
29
30     public static void alterarItemQuantidade(Item item, int quantidade) {
31         item.setQuantidade(quantidade);
32         salvarItem(item);
33     }
34
35     public static void alterarItemLimiteMinimo(Item item, int limiteMinimo) {
36         item.setLimiteMinimo(limiteMinimo);
37         salvarItem(item);
38     }
39
40     public static void alterarItem(Item item, int quantidade, int limiteMinimo) {
41         if (quantidade >= 0) {
42             item.setQuantidade(quantidade);
43         }
44
45         if (limiteMinimo >= 0) {
46             item.setLimiteMinimo(limiteMinimo);
47         }
48
49         salvarItem(item);
50     }
51
52     public static void excluirItem(Item item) {
53         item.delete();
54     }
55
56     public void consultarItem(String consulta, final RequestCallback<Item> callback) {
57         //Se a consulta for vazia, pega todos os itens do banco de dados, e coloca na memoria ram
58         if (consulta.isEmpty()) {
59             Model.find(context, Item.class, new TypeToken<ArrayList<Item>>() {
60             }.getType(), new RequestCallback<Item>() {
61             @Override
62             public void execute(ArrayList<Item> lista) throws Exception {
63                 super.execute(lista);
64
65                 itens.clear();
66                 itens.addAll(lista);
67
68                 if (callback != null) {
69                     callback.execute(lista);
70                 }
71             }
72             }, null);
73         } else {
74             //Se tiver consulta, faz a pesquisa nos itens que ja estao na memoria ram
75             try {
76                 ArrayList<Item> itensFiltrados = new ArrayList<>();
77
78                 for (Item item : itens) {
79                     if (item.getNome().contains(consulta)) {
80                         itensFiltrados.add(item);
81                     }
82                 }
83
84                 if (callback != null) {
85                     callback.execute(itensFiltrados);
86                 }
87             } catch (Exception e) {
88                 Log.e(Utils.TAG, "Erro ao consultar fornecedores: ", e);
89             }
90         }
91     }
```

```

92
93 public static void relatorioItens(ArrayList<Item> itens) {
94     for (int i = 0; i < itens.size(); i++) {
95         Item item = itens.get(i);
96         // TODO: alterar para exibicao na tela do android ou gerar pdf
97         System.out.println(
98             "Item:␣" + item.getNome() +
99             "␣Quantidade␣Disponivel:␣" + item.getQuantidade() +
100             "␣Limite␣Minimo:␣" + item.getLimiteMinimo()
101         );
102     }
103 }
104
105 public static void relatorioItensEmFalta(ArrayList<Item> itens) {
106     for (int i = 0; i < itens.size(); i++) {
107         Item item = itens.get(i);
108         if (item.getQuantidade() < item.getLimiteMinimo()) {
109             // TODO: alterar para exibicao na tela do android ou gerar pdf
110             System.out.println(
111                 "Item␣em␣Falta:␣" + item.getNome() +
112                 "␣Quantidade␣Disponivel:␣" + item.getQuantidade() +
113                 "␣Limite␣Minimo:␣" + item.getLimiteMinimo()
114             );
115         }
116     }
117 }
118
119 public static void selecionarParaEditar(Item item){
120     ControleDeItem.selecionado = item;
121 }
122
123 public static Item getSelecionado(){
124     return selecionado;
125 }
126
127 public static void deselectonar(){
128     selecionado = null;
129 }
130
131 }

```

Código 21: ControleDeItem.java

4.3.8 Controle de Produto

```

1 package ds2.equipe1.restaurante.controles;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 import ds2.equipe1.restaurante.modelos.Ingrediente;
8 import ds2.equipe1.restaurante.modelos.Item;
9 import ds2.equipe1.restaurante.modelos.Produto;
10 import java.sql.*;
11
12 public class ControleDeProduto {
13
14     private ArrayList <Produto> produtos;
15     private Context context;
16
17     public ControleDeProduto(Context context){
18         this.context = context;
19     }
20
21
22     public void carregarProdutosDoBanco(){
23         produtos = new ArrayList <Produto>();
24         int tamanhoDoCardapio = 5; //Ler o tamanho do banco
25
26         for (int i = 0; i < tamanhoDoCardapio; i++) {
27             String nome = "teste"; //Ler o nome do banco
28             //produtos.add(new Produto(nome));
29         }
30     }
31
32     public void cadastrarProduto(String nome, float preco, ArrayList <Ingrediente> ingredientes){
33         produtos.add(new Produto(context, nome, preco, ingredientes));
34     }
35
36     public void excluirProduto(String nome){
37         for (int i = 0; i < produtos.size(); i++) {
38             if (produtos.get(i).getNome() == nome) {
39                 //Remover do banco
40                 produtos.remove(i); //ver se e este mesmo o metodo de remocao
41             }
42         }
43     }
44 }

```

```

44
45     public void alterarPrecoDeProduto(String nome, float novoPreco){
46         for (int i = 0; i < produtos.size(); i++) {
47             if (produtos.get(i).getNome() == nome) {
48                 //Alterar preco no banco
49                 produtos.get(i).setPreco(novoPreco);
50             }
51         }
52     }
53
54     public Produto consultarProduto(String nome){
55         for (int i = 0; i < produtos.size(); i++) {
56             if (produtos.get(i).getNome() == nome) {
57                 return produtos.get(i);
58             }
59         }
60         return null;
61     }
62
63     public void relatorioProdutos(){
64         //Gerar pdf?
65     }
66
67     public void relatorioItensPorProduto(){
68         //Gerar pdf?
69     }
70
71 }

```

Código 22: ControleDeProduto.java

4.4 Fronteiras

4.4.1 Main Activity

```

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

```

Código 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
83             Utils.DialogCallback() {
84                 @Override
85                 public void execute(String text) {
86                     utils.setData("host", text);
87                 }
88             });
89         return true;
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) {
32                 open(CadastroFuncionario.class);
33             }
34         });
35     }
36
37     public void open(Class activity){
38         startActivity(new Intent(this, activity));
39     }
40 }

```

Código 26: MenuFuncionarios.java

4.4.5 Menu Garçom

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Toast;
8
9 public class MenuGarcom extends AppCompatActivity {

```

```

10
11     @Override
12     protected void onCreate(Bundle savedInstanceState) {
13         super.onCreate(savedInstanceState);
14         setContentView(R.layout.activity_menu_garcom);
15
16         findViewById(R.id.menu_criar_comanda).setOnClickListener(new View.OnClickListener() {
17             @Override
18             public void onClick(View v) {
19                 open(CriacaoComanda.class);
20             }
21         });
22
23         findViewById(R.id.menu_consulta_cardapio).setOnClickListener(new View.OnClickListener() {
24             @Override
25             public void onClick(View v) {
26                 open(BuscaProduto.class);
27             }
28         });
29
30         findViewById(R.id.menu_comandas_ativas).setOnClickListener(new View.OnClickListener() {
31             @Override
32             public void onClick(View v) {
33                 Toast.makeText(MenuGarcom.this, "Falta criar tela para adicionar Requisito Funcional, Diagramas etc",
34                     Toast.LENGTH_LONG).show();
35             }
36         });
37
38         public void open(Class activity){
39             startActivity(new Intent(this, activity));
40         }
41     }

```

Código 27: MenuGarcom.java

4.4.6 Menu Itens

```

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

```

Código 28: MenuItens.java

4.4.7 Menu Produtos

```

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

```

Código 29: MenuProdutos.java

4.4.8 Menu Relatório

```

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

```

Código 30: MenuRelatorio.java

4.4.9 Busca Fornecedor

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.AdapterView;
8 import android.widget.EditText;
9 import android.widget.ImageView;
10 import android.widget.ListView;
11 import android.widget.Toast;
12
13 import java.util.ArrayList;
14
15 import ds2.equipe1.restaurante.controles.ControleDeFornecedor;
16 import ds2.equipe1.restaurante.helpers.RequestCallback;
17 import ds2.equipe1.restaurante.helpers.Utils;
18 import ds2.equipe1.restaurante.listas.FornecedorAdapter;
19 import ds2.equipe1.restaurante.modelos.Fornecedor;

```

```

20
21 public class BuscaFornecedor extends AppCompatActivity {
22     private ControleDeFornecedor controleDeFornecedor;
23
24     private ListView lvFornecedores;
25     private FornecedorAdapter adapter;
26     //Fornecedores visiveis na tela
27     private ArrayList<Fornecedor> fornecedores;
28     private EditText edtProcurar;
29     private ImageView ivProcurar;
30
31     @Override
32     protected void onCreate(Bundle savedInstanceState) {
33         super.onCreate(savedInstanceState);
34         setContentView(R.layout.activity_busca_fornecedor);
35
36         init();
37
38         controleDeFornecedor = new ControleDeFornecedor(this);
39
40         fornecedores = new ArrayList<>();
41         adapter = new FornecedorAdapter(BuscaFornecedor.this, fornecedores);
42         lvFornecedores.setAdapter(adapter);
43
44         consultar("");
45     }
46
47     private void init(){
48         //pegar referencias dos componentes xml
49         lvFornecedores = (ListView) findViewById(R.id.lvFornecedores);
50         edtProcurar = (EditText) findViewById(R.id.edtProcurar);
51         ivProcurar = (ImageView) findViewById(R.id.ivProcurar);
52
53         //atribuir funcionalidades para alguns componentes
54         ivProcurar.setOnClickListener(new View.OnClickListener() {
55             @Override
56             public void onClick(View v) {
57                 String texto = edtProcurar.getText().toString();
58
59                 consultar(texto);
60             }
61         });
62
63         lvFornecedores.setOnItemClickListener(new AdapterView.OnItemClickListener() {
64             @Override
65             // no clique do item o android nos da a view, a posicao do item na lista e o ID do item
66             // (que nos configuramos) entao passamos esse ID para
67             public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
68                 ControleDeFornecedor.selecionarParaEditar(fornecedores.get(position));
69
70                 Intent intent = new Intent(BuscaFornecedor.this, CadastroFornecedor.class);
71                 intent.putExtra("alterar", true);
72                 startActivity(intent);
73             }
74         });
75     }
76
77     private void consultar(final String consulta){
78         controleDeFornecedor.consultarFornecedor(consulta, new RequestCallback<Fornecedor>(this) {
79             @Override
80             public void execute(ArrayList<Fornecedor> fornecedores) throws Exception {
81                 super.execute(fornecedores);
82
83                 BuscaFornecedor.this.fornecedores.clear();
84                 BuscaFornecedor.this.fornecedores.addAll(fornecedores);
85
86                 adapter.notifyDataSetChanged();
87             }
88         });
89     }
90
91     @Override
92     protected void onResume() {
93         //verificar se o fornecedor foi excluido para remover da tela.
94         for (int i = 0; i < fornecedores.size(); i++){
95             if (fornecedores.get(i).getId() == null){
96                 fornecedores.remove(i--);
97             }
98         }
99
100         //Quando a tela reabrir, atualizar a interface com as informacoes alteradas do item selecionado.
101         adapter.notifyDataSetChanged();
102
103         super.onResume();
104     }
105 }

```

Código 31: BuscaFornecedor.java

4.4.10 Busca Funcionário

```
1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.AdapterView;
8 import android.widget.EditText;
9 import android.widget.ImageView;
10 import android.widget.ListView;
11
12 import java.util.ArrayList;
13
14 import ds2.equipe1.restaurante.controles.ControleDeFuncionario;
15 import ds2.equipe1.restaurante.helpers.RequestCallback;
16 import ds2.equipe1.restaurante.listas.FuncionarioAdapter;
17 import ds2.equipe1.restaurante.modelos.Funcionario;
18
19 public class BuscaFuncionario extends AppCompatActivity {
20     public static ControleDeFuncionario controleDeFuncionario;
21
22     private ListView lvFuncionarios;
23     private FuncionarioAdapter adapter;
24     private ArrayList<Funcionario> funcionarios;
25     private EditText edtProcurar;
26     private ImageView ivProcurar;
27
28     @Override
29     protected void onCreate(Bundle savedInstanceState) {
30         super.onCreate(savedInstanceState);
31         setContentView(R.layout.activity_busca_funcionario);
32
33         init();
34
35         controleDeFuncionario = new ControleDeFuncionario(this);
36
37         funcionarios = new ArrayList<>();
38         adapter = new FuncionarioAdapter(BuscaFuncionario.this, funcionarios);
39         lvFuncionarios.setAdapter(adapter);
40
41         consultar("");
42     }
43     private void init(){
44         //pegar referencias dos componentes xml
45         lvFuncionarios = (ListView) findViewById(R.id.lvFuncionarios);
46         edtProcurar = (EditText) findViewById(R.id.edtProcurar);
47         ivProcurar = (ImageView) findViewById(R.id.ivProcurar);
48
49         //atribuir funcionalidades para alguns componentes
50         ivProcurar.setOnClickListener(new View.OnClickListener() {
51             @Override
52             public void onClick(View v) {
53                 String texto = edtProcurar.getText().toString();
54
55                 consultar(texto);
56             }
57         });
58
59         lvFuncionarios.setOnItemClickListener(new AdapterView.OnItemClickListener() {
60             @Override
61             //no clique do item o android nos da a view, a posicao do item na lista e o ID do item (que nos configuramos)
62             //entao passamos esse ID para
63             public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
64                 ControleDeFuncionario.selecionarParaEditar(funcionarios.get(position));
65
66                 Intent intent = new Intent(BuscaFuncionario.this, CadastroFuncionario.class);
67                 intent.putExtra("alterar", true);
68                 startActivity(intent);
69             }
70         });
71     }
72
73     private void consultar(final String consulta){
74         controleDeFuncionario.consultarFuncionario(consulta, new RequestCallback<Funcionario>(this) {
75             @Override
76             public void execute(ArrayList<Funcionario> funcionarios) throws Exception {
77                 super.execute(funcionarios);
78
79                 BuscaFuncionario.this.funcionarios.clear();
80                 BuscaFuncionario.this.funcionarios.addAll(funcionarios);
81
82                 adapter.notifyDataSetChanged();
83             }
84         });
85     }
86
87     @Override
88     protected void onResume() {
89         //verificar se o Funcionario foi excluido para remover da tela.
90         for (int i = 0; i < funcionarios.size(); i++){
```

```

91         if (funcionarios.get(i).getId() == null){
92             funcionarios.remove(i--);
93         }
94     }
95
96     //Quando a tela reabrir, atualizar a interface com as informacoes alteradas do item selecionado.
97     adapter.notifyDataSetChanged();
98
99     super.onResume();
100 }
101
102
103
104 public void onItemClick(View v){
105     startActivity(new Intent(this, CadastroFuncionario.class));
106 }
107
108
109 }

```

Código 32: BuscaFuncionario.java

4.4.11 Busca Item

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.AdapterView;
8 import android.widget.EditText;
9 import android.widget.ImageView;
10 import android.widget.ListView;
11 import android.widget.Toast;
12
13 import java.util.ArrayList;
14
15 import ds2.equipe1.restaurante.controles.ControleDeItem;
16 import ds2.equipe1.restaurante.helpers.RequestCallback;
17 import ds2.equipe1.restaurante.listas.ItemAdapter;
18 import ds2.equipe1.restaurante.modelos.Item;
19
20
21 public class BuscaItem extends AppCompatActivity {
22
23     private ControleDeItem controleDeItem;
24
25     private ListView lvItens;
26     private ItemAdapter adapter;
27
28     private ArrayList<Item> itens;
29     private EditText edtProcurar;
30     private ImageView ivProcurar;
31
32     @Override
33     protected void onCreate(Bundle savedInstanceState) {
34         super.onCreate(savedInstanceState);
35         setContentView(R.layout.activity_busca_fornecedor);
36
37         init();
38
39         controleDeItem = new ControleDeItem(this);
40
41         itens = new ArrayList<>();
42         adapter = new ItemAdapter(BuscaItem.this, itens);
43         lvItens.setAdapter(adapter);
44
45         consultar("");
46     }
47
48     private void init(){
49         //pegar referencias dos componentes xml
50         lvItens = (ListView) findViewById(R.id.lvFornecedores);
51         edtProcurar = (EditText) findViewById(R.id.edtProcurar);
52         ivProcurar = (ImageView) findViewById(R.id.ivProcurar);
53
54         //atribuir funcionalidades para alguns componentes
55         ivProcurar.setOnClickListener(new View.OnClickListener() {
56             @Override
57             public void onClick(View v) {
58                 String texto = edtProcurar.getText().toString();
59
60                 consultar(texto);
61
62                 if (itens.size() == 0){
63                     Toast.makeText(BuscaItem.this, "Nenhum resultado para " + texto + "", Toast.LENGTH_SHORT).show
64
65 ();
66

```

```

64         }
65     }
66 });
67
68     lvItens.setOnItemClickListener(new AdapterView.OnItemClickListener() {
69         @Override
70         public void onItemClick(AdapterView<?> parent, View view, int position, long id) {
71             Intent intent = new Intent(BuscaItem.this, CadastroItem.class);
72             Toast.makeText(BuscaItem.this, "" + id, Toast.LENGTH_SHORT).show();
73             intent.putExtra("id", id);
74             startActivity(intent);
75         }
76     });
77 }
78
79 private void consultar(String consulta){
80     controleDeItem.consultarItem(consulta, new RequestCallback<Item>(){
81         @Override
82         public void execute(ArrayList<Item> itens) throws Exception {
83             BuscaItem.this.itens.clear();
84             BuscaItem.this.itens.addAll(itens);
85             adapter.notifyDataSetChanged();
86
87             super.execute(itens);
88         }
89     });
90 }
91 }

```

Código 33: BuscaItem.java

4.4.12 Busca Produto

```

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

```

Código 34: BuscaProduto.java

4.4.13 Cadastro Endereço

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Button;
8 import android.widget.EditText;
9
10 import org.json.JSONObject;
11
12 import ds2.equipe1.restaurante.helpers.RequestCallback;
13 import ds2.equipe1.restaurante.modelos.Endereco;
14 import ds2.equipe1.restaurante.modelos.Model;
15
16 public class CadastroEndereco extends AppCompatActivity {
17
18     private EditText edtCEP, edtLogradouro, edtRua, edtBairro, edtCidade, edtEstado, edtNumero;
19     private Button btnCadastrar;
20
21     @Override
22     protected void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.activity_cadastro_endereco);
25     }

```



```

26     init();
27 }
28
29 private void init(){
30     btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
31     edtRua = (EditText) findViewById(R.id.edtRua);
32     edtCEP = (EditText) findViewById(R.id.edtCEP);
33     edtLogradouro = (EditText) findViewById(R.id.edtLogradouro);
34     edtBairro = (EditText) findViewById(R.id.edtBairro);
35     edtCidade = (EditText) findViewById(R.id.edtCidade);
36     edtEstado = (EditText) findViewById(R.id.edtEstado);
37     edtNumero = (EditText) findViewById(R.id.edtNumero);
38
39     btnCadastrar.setOnClickListener(new View.OnClickListener() {
40         @Override
41         public void onClick(View v) {
42
43             final Endereco endereco = new Endereco(CadastroEndereco.this, edtLogradouro.getText().toString(),
44                 edtRua.getText().toString(), Integer.parseInt(edtNumero.getText().toString()), edtBairro.getText().toString(),
45                 edtCidade.getText().toString(), edtEstado.getText().toString(), edtCEP.getText().toString());
46
47             endereco.save(new RequestCallback<Model>() {
48                 @Override
49                 public void execute(Model model) throws Exception {
50                     super.execute(model);
51
52                     Intent i = getIntent();
53                     i.putExtra("rua", endereco.getRua());
54                     i.putExtra("id_endereco", model.getId());
55                     setResult(RESULT_OK, i);
56                     finish();
57                 }
58             });
59         }
60     });
61 }

```

Código 35: CadastroEndereco.java

4.4.14 Cadastro Fornecedor

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Button;
8 import android.widget.EditText;
9
10 import ds2.equipe1.restaurante.controles.ControleDeFornecedor;
11 import ds2.equipe1.restaurante.helpers.Utills;
12 import ds2.equipe1.restaurante.modelos.Fornecedor;
13
14 public class CadastroFornecedor extends AppCompatActivity {
15     private ControleDeFornecedor controleDeFornecedor;
16     private EditText edtNome, edtCNPJ, edtEndereco, edtEmail, edtTelefone;
17     private Button btnCadastrar, btnCadastrarEndereco, btnExcluir;
18
19     private Fornecedor fornecedor;
20     private boolean novoCadastro = true;
21
22     @Override
23     protected void onCreate(Bundle savedInstanceState) {
24         super.onCreate(savedInstanceState);
25         setContentView(R.layout.activity_cadastro_fornecedor);
26
27         init();
28
29         fornecedor = new Fornecedor(this);
30         controleDeFornecedor = new ControleDeFornecedor(this);
31
32         Intent intent = getIntent();
33         if (intent.getBooleanExtra("alterar", false)){
34             novoCadastro = false;
35             carregarFornecedor();
36         }
37
38         if (novoCadastro){
39             btnExcluir.setVisibility(View.GONE);
40         } else {
41             btnCadastrar.setText("Alterar");
42             btnExcluir.setVisibility(View.VISIBLE);
43         }
44     }
45
46     private void init(){

```

```

47     edtNome = (EditText) findViewById(R.id.edtNome);
48     edtCNPJ = (EditText) findViewById(R.id.edtCNPJ);
49     edtEndereco = (EditText) findViewById(R.id.edtEndereco);
50     edtEmail = (EditText) findViewById(R.id.edtEmail);
51     edtTelefone = (EditText) findViewById(R.id.edtTelefone);
52     btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
53     btnCadastrarEndereco = (Button) findViewById(R.id.btnCadastrarEndereco);
54     btnExcluir = (Button) findViewById(R.id.btnExcluir);
55
56     btnCadastrarEndereco.setOnClickListener(new View.OnClickListener() {
57         @Override
58         public void onClick(View v) {
59             onCadastrarEnderecoClick();
60         }
61     });
62
63     btnExcluir.setOnClickListener(new View.OnClickListener() {
64         @Override
65         public void onClick(View v) {
66             if (!novoCadastro && fornecedor.getId() != null) {
67                 fornecedor.delete();
68                 fornecedor.setId(null);
69                 new Utils(CadastroFornecedor.this).toast("Fornecedor excluído!");
70                 finish();
71             }
72         }
73     });
74
75     btnCadastrar.setOnClickListener(new View.OnClickListener() {
76         @Override
77         public void onClick(View v) {
78             onCadastrarClick();
79         }
80     });
81 }
82
83 private void onCadastrarEnderecoClick(){
84     Intent intent = new Intent(this, CadastroEndereco.class);
85     startActivityForResult(intent,1);
86 }
87
88 private void onCadastrarClick(){
89     final String nome = edtNome.getText().toString();
90     final String CNPJ = edtCNPJ.getText().toString();
91     final String email = edtEmail.getText().toString();
92     final String telefone = edtTelefone.getText().toString();
93
94     /*
95         endereco.save(new RequestCallback() {
96             @Override
97             public void execute() {
98                 controleDeFornecedor.salvarFornecedor(fornecedor);
99             }
100         });*/
101
102     fornecedor.setNome(nome);
103     fornecedor.setCnpj(CNPJ);
104     fornecedor.setEmail(email);
105     fornecedor.setTelefone(telefone);
106     controleDeFornecedor.salvarFornecedor(fornecedor);
107
108     if (fornecedor.getId() == null) {
109         new Utils(this).toast("Fornecedor cadastrado!");
110     } else {
111         new Utils(this).toast("Fornecedor alterado!");
112     }
113
114     ControleDeFornecedor.deselecionar();
115     finish();
116 }
117
118 public void carregarFornecedor(){
119     if (ControleDeFornecedor.getSelecionado() != null) {
120         CadastroFornecedor.this.fornecedor = ControleDeFornecedor.getSelecionado();
121
122         edtNome.setText(fornecedor.getNome());
123         edtCNPJ.setText(fornecedor.getCnpj());
124         edtTelefone.setText(fornecedor.getTelefone());
125         edtEmail.setText(fornecedor.getEmail());
126     }
127 }
128
129 @Override
130 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
131     super.onActivityResult(requestCode, resultCode, data);
132
133     if (requestCode == 1 && resultCode == RESULT_OK){
134         if (data.hasExtra("rua")) {
135             edtEndereco.setText(data.getStringExtra("rua"));
136         }
137         /*if (data.hasExtra("id_endereco")) {
138             fornecedor.setIdEndereco(data.getIntExtra("id_endereco", -1));
139         }*/
140     }
141 }

```

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.Button;
8 import android.widget.EditText;
9
10 import ds2.equipe1.restaurante.controles.ControleDeFuncionario;
11 import ds2.equipe1.restaurante.helpers.Utills;
12 import ds2.equipe1.restaurante.modelos.Funcionario;
13
14 public class CadastroFuncionario extends AppCompatActivity {
15
16     private ControleDeFuncionario controleDeFuncionario;
17     private EditText edtNome, edtCPF, edtEndereco, edtTelefone, edtNome_de_usuario;
18     private Button btnCadastrar, btnCadastrarEndereco, btnExcluir;
19
20     private Funcionario funcionario;
21     private boolean novoCadastro = true;
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26         setContentView(R.layout.activity_cadastro_funcionario);
27
28         init();
29
30         funcionario = new Funcionario(this);
31         controleDeFuncionario = new ControleDeFuncionario(this);
32
33         Intent intent = getIntent();
34         if (intent.getBooleanExtra("alterar", false)){
35             novoCadastro = false;
36             carregarFuncionario();
37         }
38
39         if (novoCadastro){
40             btnExcluir.setVisibility(View.GONE);
41         } else {
42             btnCadastrar.setText("Alterar");
43             btnExcluir.setVisibility(View.VISIBLE);
44         }
45     }
46
47     private void init(){
48         edtNome = (EditText) findViewById(R.id.edtNome);
49         edtCPF = (EditText) findViewById(R.id.edtCPF);
50         edtNome_de_usuario = (EditText) findViewById(R.id.edtNome_de_usuario);
51         edtEndereco = (EditText) findViewById(R.id.edtEndereco);
52         edtTelefone = (EditText) findViewById(R.id.edtTelefone);
53         btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
54         btnCadastrarEndereco = (Button) findViewById(R.id.btnCadastrarEndereco);
55         btnExcluir = (Button) findViewById(R.id.btnExcluir);
56
57         btnCadastrarEndereco.setOnClickListener(new View.OnClickListener() {
58             @Override
59             public void onClick(View v) {
60                 onCadastrarEnderecoClick();
61             }
62         });
63
64         btnExcluir.setOnClickListener(new View.OnClickListener() {
65             @Override
66             public void onClick(View v) {
67                 if (!novoCadastro && funcionario.getId() != null) {
68                     funcionario.delete();
69                     funcionario.setId(null);
70                     new Utills(CadastroFuncionario.this).toast("Funcionario excluido!");
71                     finish();
72                 }
73             }
74         });
75
76         btnCadastrar.setOnClickListener(new View.OnClickListener() {
77             @Override
78             public void onClick(View v) {
79                 onCadastrarClick();
80             }
81         });
82     }

```

```

83
84 private void onCadastrarEnderecoClick(){
85     Intent intent = new Intent(this, CadastroEndereco.class);
86     startActivityForResult(intent,1);
87 }
88
89 private void onCadastrarClick(){
90     final String nome = edtNome.getText().toString();
91     final String CPF = edtCPF.getText().toString();
92     final String telefone = edtTelefone.getText().toString();
93     final String nome_de_usuario = edtNome_de_usuario.getText().toString();
94
95     funcionario.setNome(nome);
96     funcionario.setCpf(CPF);
97     funcionario.setNome_de_usuario(nome_de_usuario);
98     funcionario.setTelefone(telefone);
99     controleDeFuncionario.salvarFuncionario(funcionario);
100
101     if (funcionario.getId() == null) {
102         new Utils(this).toast("Funcionario_cadastrado!");
103     } else {
104         new Utils(this).toast("Funcionario_alterado!");
105     }
106
107     ControleDeFuncionario.deselecionar();
108     finish();
109 }
110
111 public void carregarFuncionario(){
112     if (ControleDeFuncionario.getSelecionado() != null) {
113         CadastroFuncionario.this.funcionario = ControleDeFuncionario.getSelecionado();
114
115         edtNome.setText(funcionario.getNome());
116         edtCPF.setText(funcionario.getCpf());
117         edtNome_de_usuario.setText(funcionario.getNome_de_usuario());
118         edtTelefone.setText(funcionario.getTelefone());
119     }
120 }
121
122 @Override
123 protected void onActivityResult(int requestCode, int resultCode, Intent data) {
124     super.onActivityResult(requestCode, resultCode, data);
125
126     if (requestCode == 1 && resultCode == RESULT_OK){
127         if (data.hasExtra("rua")) {
128             edtEndereco.setText(data.getStringExtra("rua"));
129         }
130         if (data.hasExtra("id_endereco")) {
131             funcionario.setIdEndereco(data.getIntExtra("id_endereco", -1));
132         }
133     }
134 }
135 }

```

Código 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
11 import ds2.equipe1.restaurante.controles.ControleDeItem;
12 import ds2.equipe1.restaurante.helpers.Utils;
13 import ds2.equipe1.restaurante.modelos.Item;
14
15 public class CadastroItem extends AppCompatActivity {
16
17     private EditText edtNome,edtQuantidade,edtUnidade,edtLimiteMinimo;
18     private Button btnCadastrar, btnExcluir;
19     private ControleDeItem controleDeItem;
20     private Item item;
21
22     private boolean novoCadastro = true;
23
24     @Override
25     protected void onCreate(Bundle savedInstanceState) {
26         super.onCreate(savedInstanceState);
27         setContentView(R.layout.activity_cadastro_item);
28
29         init();
30

```

```

31     item = new Item(this);
32     controleDeItem = new ControleDeItem(this);
33
34     Intent intent = getIntent();
35     if (intent.getBooleanExtra("alterar", false)){
36         novoCadastro = false;
37         carregarItem();
38     }
39
40     if (novoCadastro){
41         btnExcluir.setVisibility(View.GONE);
42     } else {
43         btnCadastrar.setText("Alterar");
44         btnExcluir.setVisibility(View.VISIBLE);
45     }
46 }
47
48 private void init(){
49     edtNome = (EditText) findViewById(R.id.edtNome);
50     edtQuantidade = (EditText) findViewById(R.id.edtQuantidade);
51     edtUnidade = (EditText) findViewById(R.id.edtUnidade);
52     edtLimiteMinimo = (EditText) findViewById(R.id.edtLimiteMinimo);
53     btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
54     btnExcluir = (Button) findViewById(R.id.btnExcluir);
55
56     btnExcluir.setOnClickListener(new View.OnClickListener() {
57         @Override
58         public void onClick(View v) {
59             if (!novoCadastro && item.getId() != null) {
60                 item.delete();
61                 item.setId(null);
62                 new Utils(CadastroItem.this).toast("Item excluído!");
63                 finish();
64             }
65         }
66     });
67
68     btnCadastrar.setOnClickListener(new View.OnClickListener() {
69         @Override
70         public void onClick(View v) {
71             onCadastrarClick();
72         }
73     });
74 }
75
76 private void onCadastrarClick() {
77     final String nome = edtNome.getText().toString();
78     final int quantidade = Integer.parseInt(edtQuantidade.getText().toString());
79     final String unidade = edtUnidade.getText().toString();
80     final int limite = Integer.parseInt(edtLimiteMinimo.getText().toString());
81
82     item.setNome(nome);
83     item.setQuantidade(quantidade);
84     item.setUnidade(unidade);
85     item.setLimiteMinimo(limite);
86
87     controleDeItem.salvarItem(item);
88
89     if (item.getId() == null) {
90         new Utils(this).toast("Item cadastrado!");
91     } else {
92         new Utils(this).toast("Item alterado!");
93     }
94 }
95
96 }
97
98 private void carregarItem(){
99     if (ControleDeItem.getSelecionado() != null) {
100         CadastroItem.this.item = ControleDeItem.getSelecionado();
101
102         edtNome.setText(item.getNome());
103         edtQuantidade.setText(item.getQuantidade());
104         edtUnidade.setText(item.getUnidade());
105         edtLimiteMinimo.setText(item.getLimiteMinimo());
106     }
107 }
108
109 }
110 }

```

Código 38: CadastroItem.java

4.4.17 Cadastro Produto

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Context;

```

```

4 import android.content.Intent;
5 import android.support.v7.app.AppCompatActivity;
6 import android.os.Bundle;
7 import android.view.View;
8 import android.widget.Button;
9 import android.widget.EditText;
10 import android.widget.ImageButton;
11
12 import java.util.ArrayList;
13
14 import ds2.equipe1.restaurante.controles.ControleDeProduto;
15 import ds2.equipe1.restaurante.helpers.Utills;
16 import ds2.equipe1.restaurante.modelos.Ingrediente;
17 import ds2.equipe1.restaurante.modelos.Produto;
18
19 public class CadastroProduto extends AppCompatActivity {
20
21     private ControleDeProduto controleDeProduto;
22     private EditText edtNome, edtPreco;
23     private Button btnCadastrar, btnCancel;
24     private ImageButton btnAddIngrediente;
25     private ArrayList<Ingrediente> ingredientes = new ArrayList<Ingrediente>();
26     private Context context;
27
28     @Override
29     protected void onCreate(Bundle savedInstanceState) {
30         super.onCreate(savedInstanceState);
31         setContentView(R.layout.activity_cadastro_produto);
32         //this.context = ??? PEGAR CONTEXT0;
33
34         init();
35
36         controleDeProduto = new ControleDeProduto(this);
37     }
38
39     private void init(){
40         edtNome = (EditText) findViewById(R.id.edtNome);
41         edtPreco = (EditText) findViewById(R.id.edtPreco);
42         btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
43         btnCancel = (Button) findViewById(R.id.btnCancel);
44         btnAddIngrediente = (ImageButton) findViewById(R.id.btnAddIngrediente);
45
46         btnCadastrar.setOnClickListener(new View.OnClickListener() {
47             @Override
48             public void onClick(View v) {
49                 onCadastrarClick();
50             }
51         });
52
53         btnCancel.setOnClickListener(new View.OnClickListener() {
54             @Override
55             public void onClick(View v) {
56                 onCancelClick();
57             }
58         });
59
60         btnAddIngrediente.setOnClickListener(new View.OnClickListener() {
61             @Override
62             public void onClick(View v) {
63                 onAddIngredienteClick();
64             }
65         });
66     }
67
68     private void onCadastrarClick(){
69         final String nome = edtNome.getText().toString();
70         final Float preco = Float.parseFloat(edtPreco.getText().toString());
71
72         Produto produto = new Produto(context,nome,preco,ingredientes);
73
74         new Utills(this).toast("Produto_cadastrado!");
75         finish();
76     }
77
78     private void onCancelClick(){
79         finish();
80     }
81
82     private void onAddIngredienteClick(){
83         new Utills(this).selectPopup("Cadastrar_ingrediente", null);
84
85         //Intent intent = new Intent(this, SelecionarIngredientes.class);
86         //startActivityForResult(intent,1);
87
88         //Implementar lista de selecao de ingredientes
89     }
90 }
91

```

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 }
```

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.modelos.Fornecedor;
15
16 public class FornecedorAdapter extends BaseAdapter {
17     private Context context;
18     private ArrayList<Fornecedor> fornecedores;
19
20     public FornecedorAdapter(Context context, ArrayList<Fornecedor> fornecedores) {
21         this.context = context;
22         this.fornecedores = fornecedores;
23     }
24
25     @Override
26     public int getCount() {
27         return fornecedores.size();
28     }
29
30     @Override
31     public Fornecedor getItem(int position) {
32         return fornecedores.get(position);
33     }
34
35     @Override
36     public long getItemId(int position) {
37         return fornecedores.get(position).getId();
38     }
39
40     @Override
41     public View getView(int position, View root, ViewGroup parent) {
42         if (root == null) {
43             LayoutInflater vi = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
44             root = vi.inflate(R.layout.item_fornecedor, parent, false);
45         }
46
47         TextView edtNome, edtCNPJ, edtTelefone;
48         edtNome = (TextView) root.findViewById(R.id.tvNome);
49         edtCNPJ = (TextView) root.findViewById(R.id.tvCNPJ);
50         edtTelefone = (TextView) root.findViewById(R.id.tvTelefone);
51
52         Fornecedor fornecedor = fornecedores.get(position);
53
54         edtNome.setText(fornecedor.getNome());
55         edtCNPJ.setText(fornecedor.getCnpj());
56         edtTelefone.setText(fornecedor.getTelefone());
57
58         return root;
59     }
60 }

```

Código 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.modelos.Funcionario;
14
15 public class FuncionarioAdapter extends BaseAdapter {
16     private Context context;
17     private ArrayList<Funcionario> funcionarios;
18
19     public FuncionarioAdapter(Context context, ArrayList<Funcionario> funcionarios) {
20         this.context = context;
21         this.funcionarios = funcionarios;
22     }
23
24     @Override
25     public int getCount() {
26         return funcionarios.size();
27     }
28
29     @Override
30     public Funcionario getItem(int position) {
31         return funcionarios.get(position);
32     }
33
34     @Override
35     public long getItemId(int position) {
36         return funcionarios.get(position).getId();
37     }
38
39     @Override
40     public View getView(int position, View root, ViewGroup parent) {
41         if (root == null) {
42             LayoutInflater vi = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
43             root = vi.inflate(R.layout.item_funcionario, parent, false);
44         }
45
46         TextView edtNome, edtCPF, edtTelefone;
47         edtNome = (TextView) root.findViewById(R.id.tvNome);
48         edtCPF = (TextView) root.findViewById(R.id.tvCPF);
49         edtTelefone = (TextView) root.findViewById(R.id.tvTelefone);
50
51         Funcionario funcionario = funcionarios.get(position);
52
53         edtNome.setText(funcionario.getNome());
54         edtCPF.setText(funcionario.getCpf());
55         edtTelefone.setText(funcionario.getTelefone());
56
57         return root;
58     }
59 }

```

Código 46: FuncionarioAdapter.java

4.5.3 Item Adapter

```

1 package ds2.equipe1.restaurante.listas;
2
3 import android.content.Context;
4 import android.view.LayoutInflater;
5 import android.view.View;
6 import android.view.ViewGroup;
7 import android.widget.BaseAdapter;
8 import android.widget.EditText;
9 import android.widget.TextView;
10
11 import java.util.ArrayList;
12
13 import ds2.equipe1.restaurante.R;
14 import ds2.equipe1.restaurante.modelos.Item;
15
16 public class ItemAdapter extends BaseAdapter {
17     private Context context;
18     private ArrayList<Item> itens;
19
20     public ItemAdapter(Context context, ArrayList<Item> fornecedores) {
21         this.context = context;
22         this.itens = fornecedores;
23     }

```

```

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

```

Código 47: ItemAdapter.java

4.5.4 Produto Adapter

```

1  package ds2.equipe1.restaurante.listas;
2
3  import android.content.Context;
4  import android.view.LayoutInflater;
5  import android.view.View;
6  import android.view.ViewGroup;
7  import android.widget.BaseAdapter;
8  import android.widget.TextView;
9
10 import java.util.ArrayList;
11
12 import ds2.equipe1.restaurante.R;
13 import ds2.equipe1.restaurante.modelos.Produto;
14
15 public class ProdutoAdapter extends BaseAdapter {
16     private Context context;
17     private ArrayList<Produto> produtos;
18
19     public ProdutoAdapter(Context context, ArrayList<Produto> produtos) {
20         this.context = context;
21         this.produtos = produtos;
22     }
23
24     @Override
25     public int getCount() {
26         return produtos.size();
27     }
28
29     @Override
30     public Produto getItem(int position) {
31         return produtos.get(position);
32     }
33
34     @Override
35     public long getItemId(int position) {
36         return produtos.get(position).getId();
37     }
38
39     @Override
40     public View getView(int position, View root, ViewGroup parent) {
41         if (root == null) {
42             LayoutInflater vi = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
43             root = vi.inflate(R.layout.item_produto, parent, false);
44         }
45
46         TextView edtNome, edtPreco;
47         edtNome = (TextView) root.findViewById(R.id.tvNome);
48         edtPreco = (TextView) root.findViewById(R.id.tvPreco);
49

```

```

50     Produto produto = produtos.get(position);
51
52     edtNome.setText(produto.getNome());
53     edtPreco.setText(Float.toString(produto.getPreco()));
54
55     return root;
56 }
57 }

```

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
7  import com.androidquery.AQuery;
8  import com.androidquery.callback.AjaxCallback;
9  import com.androidquery.callback.AjaxStatus;
10 import com.google.gson.Gson;
11 import com.google.gson.internal.Excluder;
12 import com.google.gson.reflect.TypeToken;
13 import com.google.gson.stream.MalformedJsonException;
14
15 import org.json.JSONException;
16 import org.json.JSONObject;
17
18 import java.lang.reflect.Type;
19 import java.util.ArrayList;
20 import java.util.HashMap;
21 import java.util.Map;
22 import java.util.Objects;
23
24 import ds2.equipe1.restaurante.modelos.Fornecedor;
25 import ds2.equipe1.restaurante.modelos.Ingrediente;
26 import ds2.equipe1.restaurante.modelos.Model;
27 import ds2.equipe1.restaurante.modelos.Produto;
28
29 /**
30  * Created by Fernando on 07/04/2016.
31  */
32 public class ServerRequest {
33     private static final String TAG = Utils.TAG;
34
35     public enum Action {
36         INDEX, SAVE, DELETE, GET, FIND
37     }
38
39     private Context context;
40
41     public ServerRequest(Context context) {
42         this.context = context;
43     }
44
45     public void sendRequest(String controller, Action action, String data, final RequestCallback<JSONObject> callback){
46         Map<String, Object> params = getAuthParams();
47         params.put("data", data);
48
49         new AQuery(context).ajax(buildURL(controller, action), params, JSONObject.class, new AjaxCallback<JSONObject>(){
50             {
51                 @Override
52                 public void callback(String url, JSONObject json, AjaxStatus status) {
53                     try {
54                         Log.w(TAG, "[" + url + "]: " + json.toString());
55                         if (callback != null){
56                             callback.execute(json);
57                         }
58                     } catch (Exception e){
59                         Log.e(TAG, "Erro no processamento de dados: ", e);
60                         Log.e(TAG, "AjaxStatus: " + status.getError() + " " + status.getStatusCode() + " " + status.
61                             getMessage());
62                     }
63                     super.callback(url, json, status);
64                 }
65             }
66
67             @Override
68             public void failure(int code, String message) {
69                 Log.e(TAG, "Falha de conexao: " + message + " " + code);
70                 super.failure(code, message);
71             }
72         }).timeout(5000);

```

```

70     }
71
72     private Map<String, Object> getAuthParams(){
73         Map<String, Object> params = new HashMap<>();
74         //TODO: Remover padrao
75         params.put("usuario", new Utils(context).getData("usuario", "user.default"));
76         params.put("senha", new Utils(context).getData("senha", "senha.default"));
77         return params;
78     }
79
80     private String buildURL(String controller, Action action){
81         return new Utils(context).getData("host", "http://192.168.0.105/restaurante/") + controller.toLowerCase() + "/"
82         + action.name().toLowerCase() + "/";
83     }
84 }

```

Código 49: ServerRequest.java

4.6.2 Request Callback

```

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

```

Código 50: RequestCallback.java

4.6.3 Utilitários

```

1  package ds2.equipe1.restaurante.helpers;
2
3  import android.app.ProgressDialog;
4  import android.content.Context;
5  import android.content.DialogInterface;
6  import android.content.SharedPreferences;
7  import android.support.v7.app.AlertDialog;
8  import android.view.LayoutInflater;
9  import android.widget.EditText;
10 import android.widget.LinearLayout;
11 import android.widget.Spinner;
12 import android.widget.Toast;
13
14 import ds2.equipe1.restaurante.R;
15

```

```

16 public class Utils {
17     public static final String TAG = "Restaurante";
18     private Context context;
19     private static ProgressDialog progress;
20
21     public Utils(Context context) {
22         this.context = context;
23     }
24
25     public void deleteData(String key){
26         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
27         SharedPreferences.Editor e = settings.edit();
28         e.remove(key);
29         e.commit();
30     }
31
32     public float getData(String key, float defValue) {
33         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
34         return settings.getFloat(key, defValue);
35     }
36
37     public String getData(String key, String defValue) {
38         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
39         return settings.getString(key, defValue);
40     }
41
42     public int getData(String key, int defValue) {
43         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
44         return settings.getInt(key, defValue);
45     }
46
47     public long getData(String key, long defValue) {
48         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
49         return settings.getLong(key, defValue);
50     }
51
52     public boolean getData(String key, boolean defValue) {
53         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
54         return settings.getBoolean(key, defValue);
55     }
56
57     public void setData(String key, float value) {
58         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
59         settings.putFloat(key, value);
60         settings.commit();
61     }
62
63     public void setData(String key, String value) {
64         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
65         settings.putString(key, value);
66         settings.commit();
67     }
68
69     public void setData(String key, int value) {
70         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
71         settings.putInt(key, value);
72         settings.commit();
73     }
74
75     public void setData(String key, long value) {
76         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
77         settings.putLong(key, value);
78         settings.commit();
79     }
80
81     public void setData(String key, boolean value) {
82         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
83         settings.putBoolean(key, value);
84         settings.commit();
85     }
86
87     public void clearData(){
88         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
89         SharedPreferences.Editor editor = settings.edit();
90         editor.clear();
91         editor.commit();
92     }
93
94     public static void launchProgress(Context context){
95         launchProgress(context, null, "Carregando...");
96     }
97
98     public static void launchProgress(Context context, String title, String text) {
99         launchProgress(context, title, text, null);
100     }
101
102     public static void launchProgress(Context context, String title, String text, DialogInterface.OnDismissListener
        func) {
103         progress = ProgressDialog.show(context, title, text, true);
104         progress.setCancelable(true);
105         if (func != null) progress.setOnDismissListener(func);
106     }
107
108     public static void dismissProgress() {

```

```

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

```

Código 51: Utils.java

5 Codificação referente ao Servidor (PHP)

5.1 Modelos

5.1.1 Comanda

```
1 <?php
2
3 use Phalcon\Mvc\Model;
4 use Phalcon\Mvc\Model\Message;
5
6 //TODO: Remover total de pedidos
7 class Comanda extends Model {
8
9     public $id;
10    public $data;
11    public $ativa;
12    public $nome;
13    public $pedidos;
14
15    public function getSchema() {
16        return "public";
17    }
18 }
19
20 ?>
```

Código 52: Comanda.php

5.1.2 Compra

```
1 <?php
2
3 use Phalcon\Mvc\Model;
4 use Phalcon\Mvc\Model\Message;
5
6
7 class Compra extends Model {
8
9     public $id;
10    public $item;
11    public $quantidade;
12    public $preco;
13    public $data;
14
15    public function getSchema() {
16        return "public";
17    }
18 }
19
20 ?>
```

Código 53: Compra.php

5.1.3 Endereço

```
1 <?php
2
3 use Phalcon\Mvc\Model;
4 use Phalcon\Mvc\Model\Message;
5
6
7 class Endereco extends Model {
8
9     public $id;
10    public $logradouro;
11    public $rua;
12    public $numero;
13    public $bairro;
14    public $cidade;
15    public $estado;
16    public $cep;
17
18    /*public function beforeSave(){
19        if (!$rua || !$logradouro || !$bairro){
20            return false;
21        }
22    }*/
```



```

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/restaurantefornecedor/save/?data={%22nome%22:%22Fernando%20Messias%22,%20%22telefone
4      %22:%20%2279%2098833848%22,%20%22cnpj%22:%20%2200-000-0000-00%22,%20%22email%22:%22fernandoxlr@live.com
5      %22,%20%22id_endereco%22:1}
6
7  use Phalcon\Mvc\Model;
8  use Phalcon\Mvc\Model\Message;
9
10 class Fornecedor extends Model {
11
12     public $id;
13     public $nome;
14     public $telefone;
15     public $cnpj;
16     public $email;
17     public $id_endereco;
18
19     public function initialize(){
20         $this->belongsTo("id_endereco", "Endereco", "id");
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

```
1 <?php
2
3 use Phalcon\Mvc\Model;
4 use Phalcon\Mvc\Model\Message;
5
6 class Ingrediente extends Model {
7     public $id_item;
8     public $id_producto;
9     public $quantidade;
10
11     public function initialize(){
12         $this->belongsTo("id_producto", "Produto", "id");
13         $this->belongsTo("id_item", "Item", "id");
14     }
15
16     public function getSchema() {
17         return "public";
18     }
19 }
```

Código 57: Ingrediente.php

5.1.7 Item

```
1 <?php
2
3 use Phalcon\Mvc\Model;
4 use Phalcon\Mvc\Model\Message;
5
6
7 class Item extends Model {
8
9     public $id;
10    public $nome;
11    public $unidade;
12    public $quantidade;
13    public $limite;
14
15    public function getSchema() {
16        return "public";
17    }
18 }
19
20 ?>
```

Código 58: Item.php

5.1.8 Produto

```
1 <?php
2
3 use Phalcon\Mvc\Model;
4 use Phalcon\Mvc\Model\Message;
5
6
7 class Produto extends Model {
8
9     public $id;
10    public $nome;
11    public $preco;
12
13    public function initialize(){
14        $this->hasMany("id", "Ingrediente", "id_producto");
15    }
16
17    public function getSchema() {
18        return "public";
19    }
20 }
21
22 ?>
```

Código 59: Produto.php

5.2 Controles

5.2.1 Controlador Base

```
1 <?php
2
3 use Phalcon\Mvc\Controller;
4 use Phalcon\Mvc\Dispatcher;
5
6 class BaseController extends Controller {
7
8     public $debugArray;
9     public $lastDebugKey = 1;
10
11     /**
12      * This action is available for multiple controllers
13      */
14     public function redis() {
15         return $this->getDi()->getRedis();
16     }
17
18     // After route executed event. Convert everything to JSON response
19     public function afterExecuteRoute(Dispatcher $dispatcher) {
20         // If the response was sent before (404 example below)
21         if ($this->response->isSent()) {
22             return;
23         }
24
25         $data = $dispatcher->getReturnedValue();
26
27         /* Set global params if is not set in controller/action */
28         return $this->sendContent(200, "", $data);
29     }
30
31     public function sendContent($statusCode, $statusMessage, $content) {
32         if ($this->response->isSent()) {
33             return;
34         }
35
36         if ($content == null) {
37             $content = array();
38         } else if (!is_array($content)) {
39             $content = array($content);
40         }
41
42         $content['success'] = isset($content['success']) ? $content['success'] : true;
43
44         if ($this->debugArray) {
45             $content['debug'] = $this->debugArray;
46         }
47
48         $this->response->setContentType('application/json', 'UTF-8');
49         $this->response->setHeader('Access-Control-Allow-Headers', 'X-Requested-With');
50         $this->response->setStatusCode($statusCode, $statusMessage);
51         $this->response->setContent(json_encode($content));
52         return $this->response->send();
53     }
54
55
56
57     public function notFoundAction() {
58         $this->response->setStatusCode(404, 'Not Found');
59         $this->response->setContentType('application/json', 'UTF-8');
60         $this->response->setContent(json_encode(array("success" => false, "message" => "This endpoint doesn't exist.")));
61         implode(",", $this->request->getQuery());
62         return $this->response->send();
63     }
64
65     public function buildErrorReturn($message) {
66         return array("success" => false, "message" => $message);
67     }
68
69     public function buildReturn($data, $key = 'data') {
70         return array("success" => true, $key => $data);
71     }
72
73     public function debug($obj, $key = 0) {
74         if (!$this->debugArray) {
75             $this->debugArray = array();
76         }
77         if (!$key) {
78             $key = $this->lastDebugKey++;
79         }
80         $this->debugArray[$key] = $obj;
81     }
82
83     public function saveAction(){
84         try {
85             $model = ucfirst($this->router->getControllerName());
86
87             $instance = new $model();
88         }
89     }
90 }
```

```

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 {

```

```

20         //return ["success" => false, "errors" => ["nada nao"]];
21         $errors = [];
22         foreach ($endereco->getMessages() as $error) {
23             $errors[] = $error->getMessage();
24         }
25         return ["success" => false, "errors" => $errors];
26     }
27
28     return ["id" => $fornecedor->id];
29
30 }
31
32 public function findAction(){
33     try {
34         $fornecedores = Fornecedor::find();
35         $arr = [];
36         foreach ($fornecedores as $fornecedor){
37             $fornecedor->endereco;
38             $fornecedor = get_object_vars($fornecedor);
39             $arr[] = $fornecedor;
40         }
41         return ["data" => $arr];
42     } catch (Exception $e){
43         print_r($e);
44     }
45     exit;
46 }
47 }

```

Código 62: FornecedorController.php

5.2.4 Controlador de Funcionário

```

1  <?php
2
3  class FuncionarioController extends BaseController {
4      public function saveAction(){
5          $funcionario_arr = json_decode($this->request->get("data"), true);
6          $endereco_arr = $funcionario_arr["endereco"];
7          $endereco = new Endereco();
8          $funcionario = new Funcionario();
9
10         if ($endereco->save($endereco_arr)){
11             $funcionario->id_endereco = $endereco->id;
12             if (!$funcionario->save($funcionario_arr)){
13                 $errors = [];
14                 foreach ($funcionario->getMessages() as $error) {
15                     $errors[] = $error->getMessage();
16                 }
17                 return ["success" => false, "errors" => $errors];
18             }
19         } else {
20             //return ["success" => false, "errors" => ["nada nao"]];
21             $errors = [];
22             foreach ($endereco->getMessages() as $error) {
23                 $errors[] = $error->getMessage();
24             }
25             return ["success" => false, "errors" => $errors];
26         }
27
28         return ["id" => $funcionario->id];
29     }
30
31
32     public function findAction(){
33         try {
34             $funcionarios = Funcionario::find();
35             $arr = [];
36             foreach ($funcionarios as $funcionario){
37                 $endereco = $funcionario->endereco;
38                 $funcionario = get_object_vars($funcionario);
39                 /*if ($endereco){
40                     $funcionario["endereco"] = $endereco;
41                 }*/
42                 $arr[] = $funcionario;
43             }
44             return ["data" => $arr];
45         } catch (Exception $e){
46             print_r($e);
47         }
48         exit;
49     }
50 }

```

Código 63: FuncionarioController.php

5.2.5 Controlador de Item

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

Código 64: ItemController.php

5.2.6 Controlador de Produto

```
1
2
3 <?php
4
5 class ProdutoController extends BaseController {
6     public function saveAction(){
7         try {
8             $produto = new Produto();
9             $data = json_decode($this->request->get("data"), true);
10
11             $produto->nome = $data["nome"];
12             $produto->preco = $data["preco"];
13             $ingredientesArr = $data["ingredientes"];
14
15             if ($produto->save($data)){
16                 foreach ($ingredientesArr as $ingredienteArr) {
17                     $item = $ingredienteArr["item"];
18                     $ingrediente = new Ingrediente();
19                     $ingrediente->id_item = $item["id"];
20                     $ingrediente->id_produto = $produto->id;
21                     $ingrediente->quantidade = $ingredienteArr["quantidade"];
22                     if (!$ingrediente->save()){
23                         $errors = [];
24                         foreach ($ingrediente->getMessages() as $error) {
25                             $errors[] = $error->getMessage();
26                         }
27                         return ["success" => false, "errors" => $errors];
28                     }
29                 }
30                 return ["id" => $produto->id];
31             } else {
32                 $errors = [];
33                 foreach ($produto->getMessages() as $error) {
34                     $errors[] = $error->getMessage();
35                 }
36                 return ["success" => false, "errors" => $errors];
37             }
38         } catch (Exception $e){
39             print_r($e);
40         }
41     }
42
43     public function findAction(){
44         try {
45             $produtos = Produto::find();
46             $arr = [];
47             foreach ($produtos as $produto){
48                 $ingredientes = $produto->getIngrediente();
49                 $ingredientes_arr = $ingredientes->toArray();
50                 for ($i = 0; $i < count($ingredientes_arr); $i++){
51                     $ingredientes_arr[$i]["item"] = $ingredientes[$i]->getItem();
52                 }
53                 $produto = get_object_vars($produto);
54                 if ($ingredientes){
55                     $produto["ingredientes"] = $ingredientes_arr;
56                 }
57                 $arr[] = $produto;
58             }
59             return ["data" => $arr];
60         } catch (Exception $e){
61             print_r($e);
62         }
63         exit;
64     }
65 }
```

Código 65: ProdutoController.php

5.3 Banco de Dados

```

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

```

```

94     id_endereco integer,
95     CONSTRAINT pk_fornecedor PRIMARY KEY (id),
96     CONSTRAINT fk_id_endereco FOREIGN KEY (id_endereco) REFERENCES endereco (id),
97     CONSTRAINT unique_cnpj UNIQUE (cnpj)
98 );
99
100 CREATE TABLE compra (
101     id serial,
102     id_item integer,
103     id_fornecedor integer,
104     quantidade integer,
105     preco float,
106     data timestamp,
107     CONSTRAINT pk_compra PRIMARY KEY (id),
108     CONSTRAINT fk_compra_item FOREIGN KEY (id_item) REFERENCES item (id),
109     CONSTRAINT fk_compra_fornecedor FOREIGN KEY (id_fornecedor) REFERENCES fornecedor (id)
110 );

```

Código 66: CriarBanco.sql

6 Diagramas

6.1 Diagrama de Classes

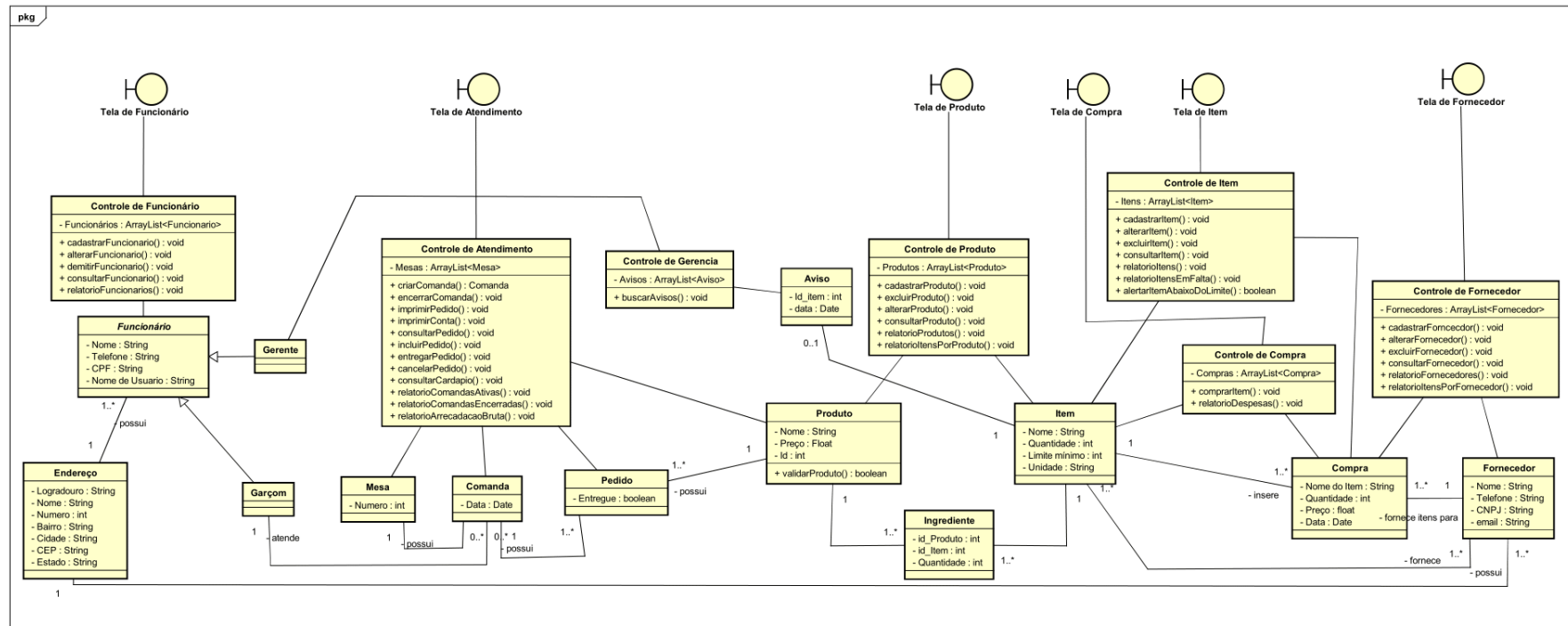


Figura 1: Diagrama de Classes

6.2 Diagrama de Sequência

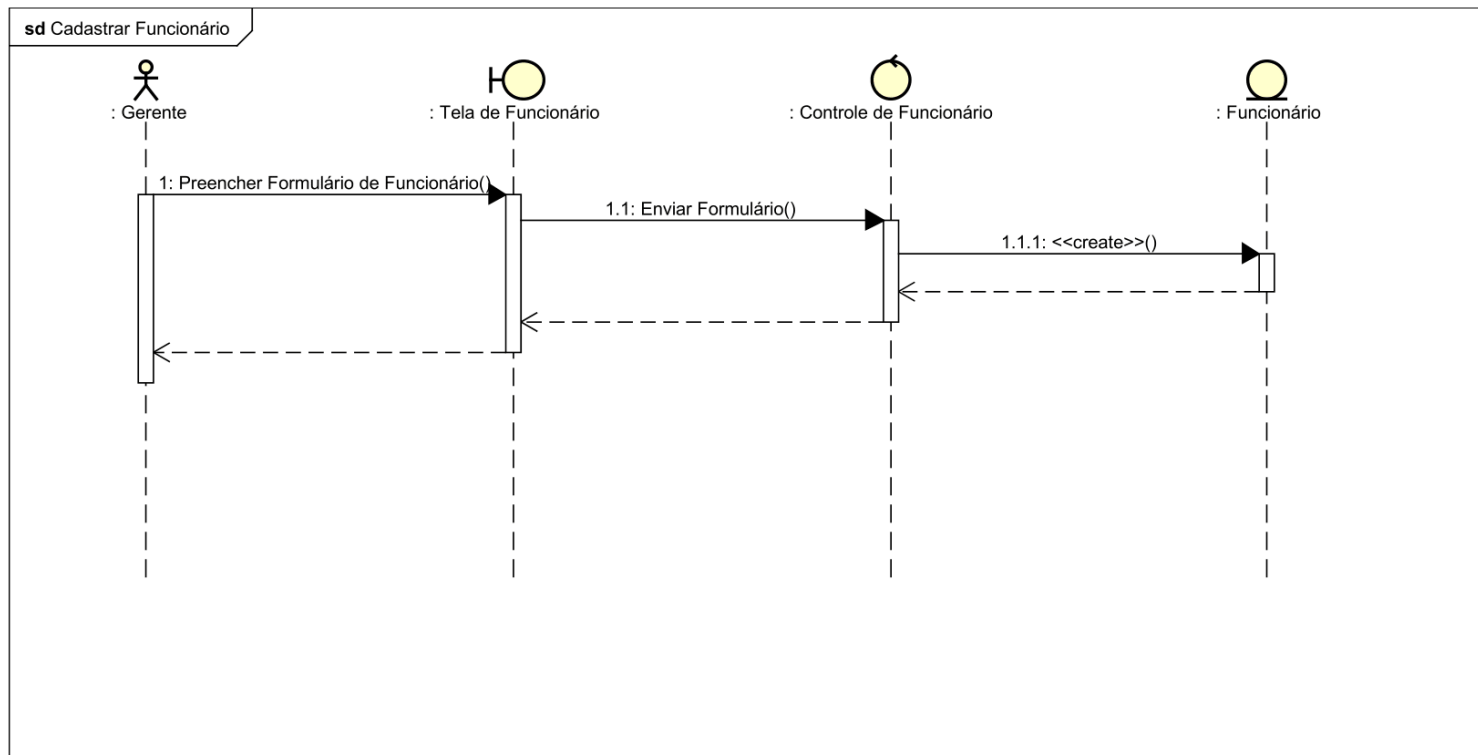


Figura 2: Cadastrar Funcionário

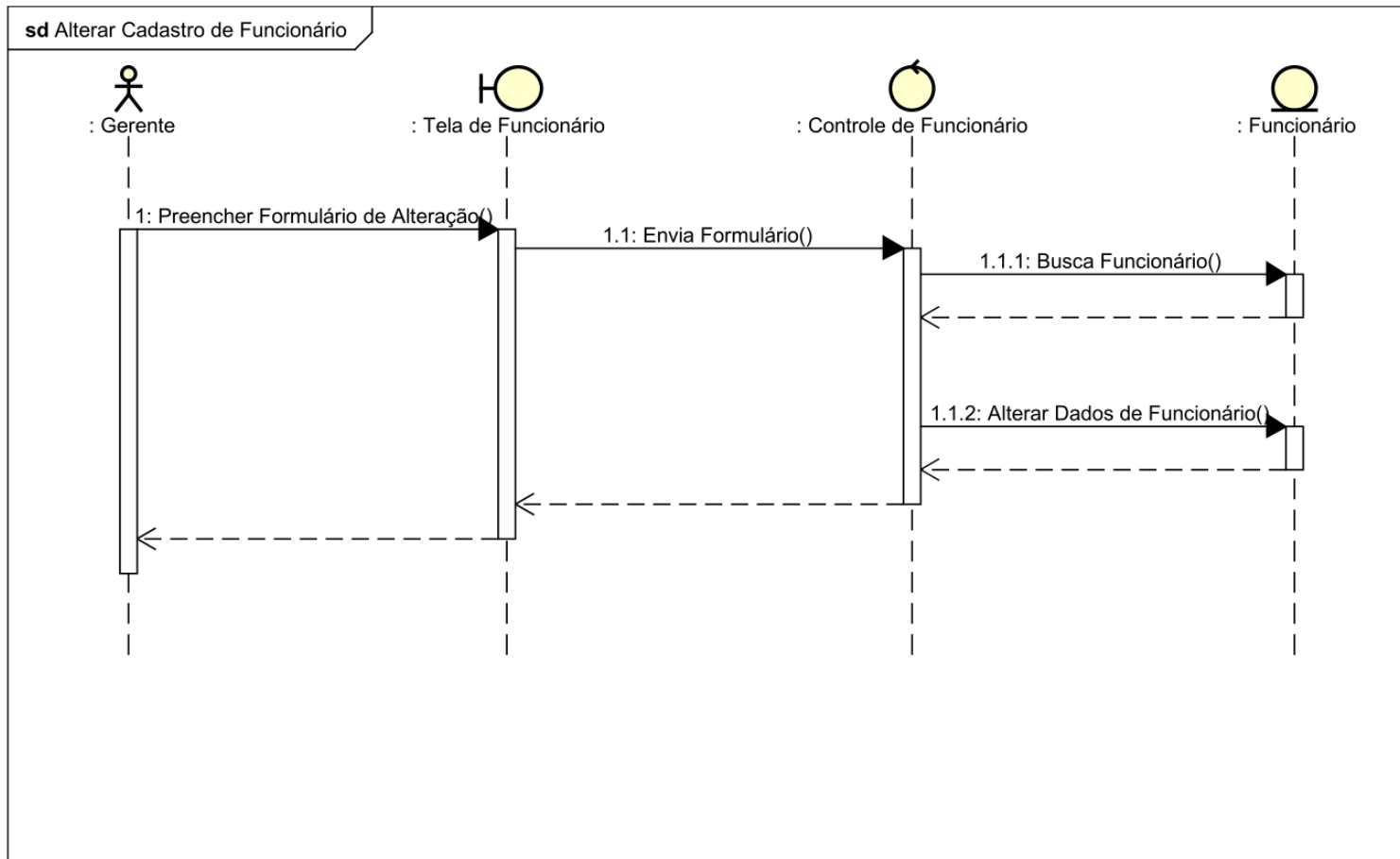


Figura 3: Alterar Cadastro de Funcionário

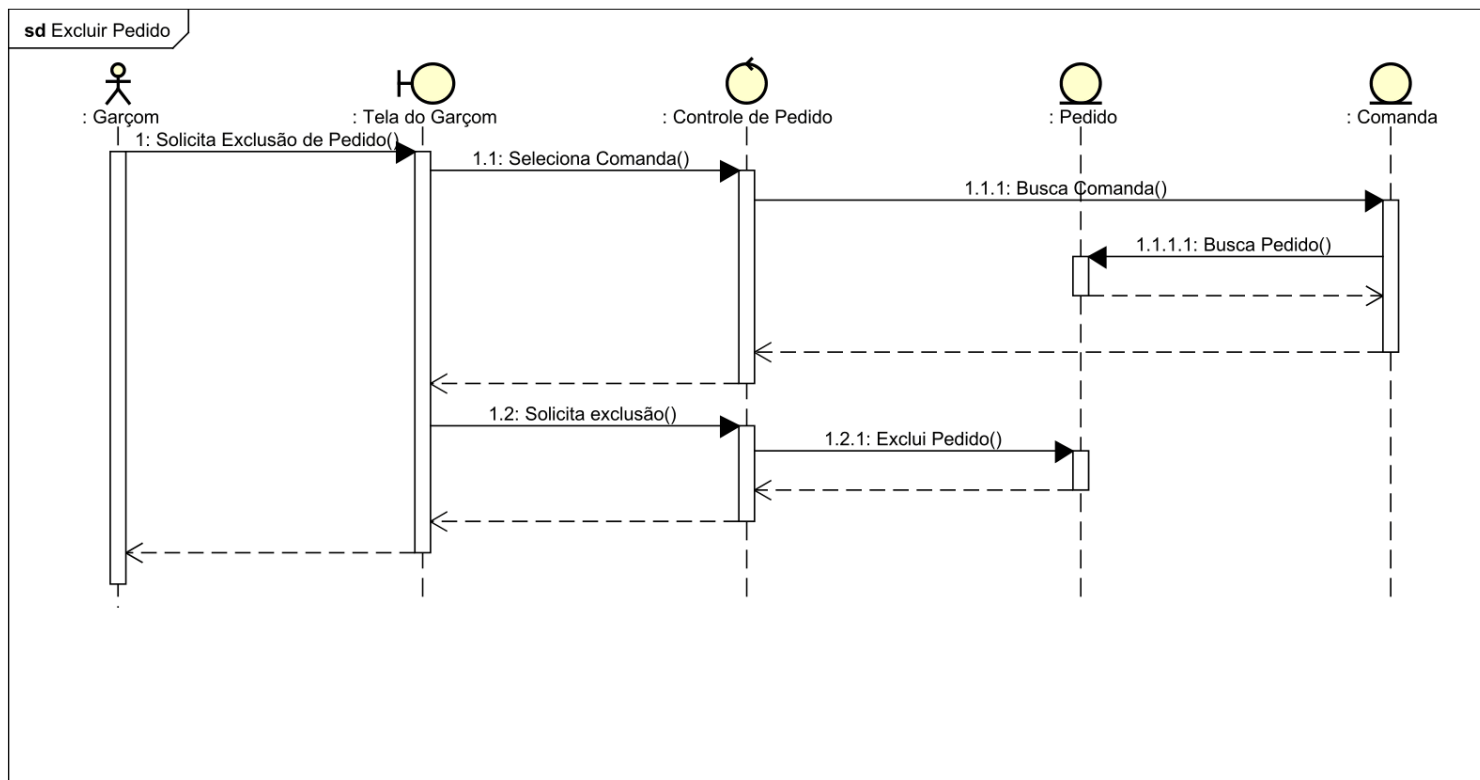


Figura 4: Excluir Pedido

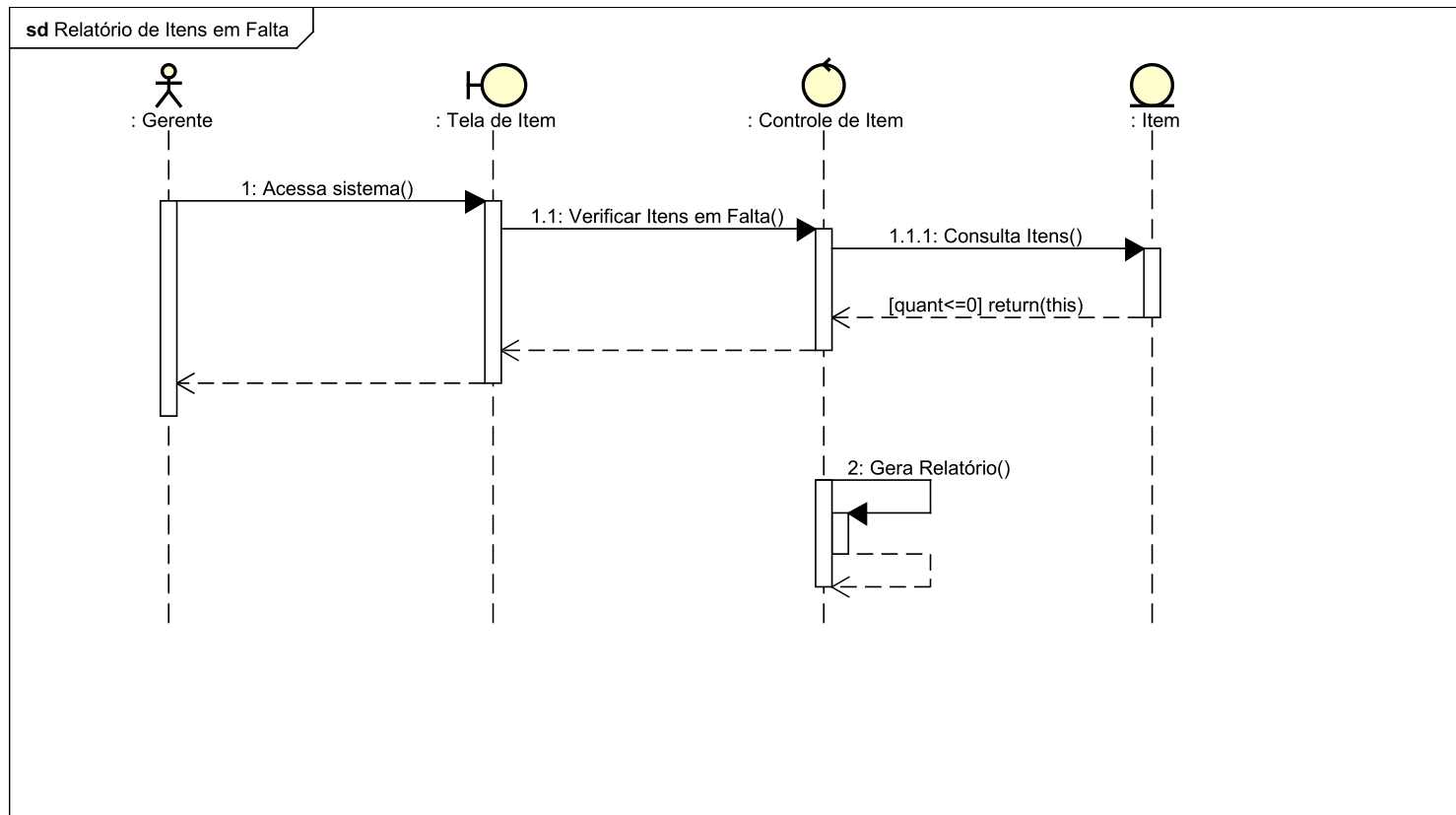


Figura 5: Relatório de Itens em Falta

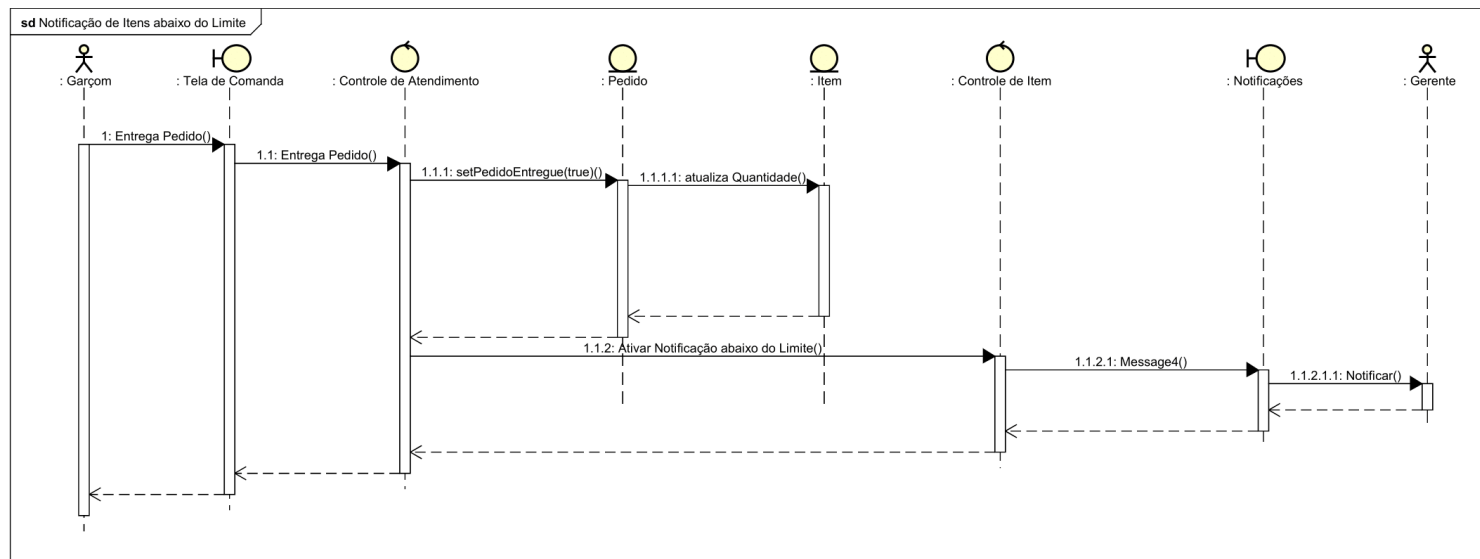


Figura 6: Notificação de Itens abaixo do limite

6.3 Diagrama de Casos de Uso

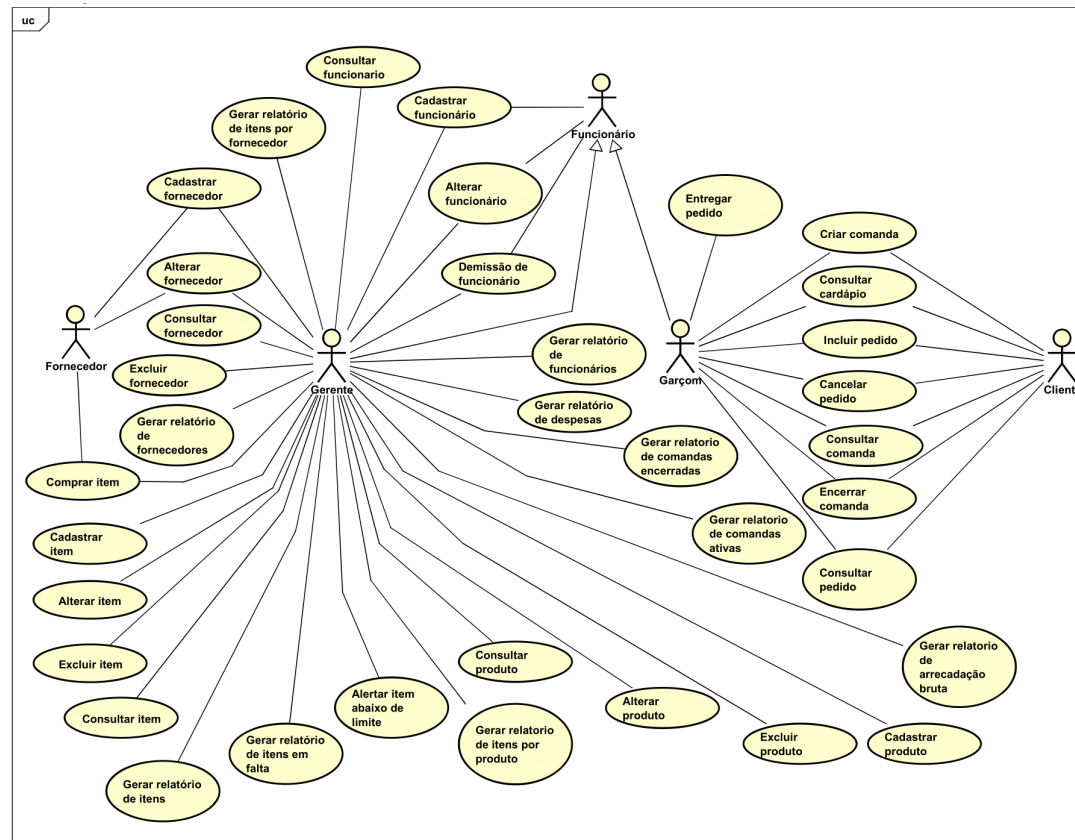


Figura 7: Diagrama de Casos de Uso