



**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	5
1.1 Propósito do Documento	5
1.2 Escopo do Produto	5
1.3 Definições e Abreviações	5
1.3.1 Definições	5
1.3.2 Abreviações	5
1.4 Referências	5
1.5 Visão Geral do Restante do Documento	6
1.6 Descrição Geral	6
1.6.1 Perspectiva do Produto	6
1.6.2 Funções do Produto	6
1.6.3 Características do Usuário	6
1.6.4 Restrições Gerais	6
1.6.5 Suposições e Dependências	6
1.7 Requisitos específicos	7
1.7.1 Requisitos Funcionais	7
1.7.2 Requisitos Não Funcionais	9
2 Plano de Projeto	11
3 Casos de Uso	12
4 Código do Projeto	15
4.1 Modelos	15
4.1.1 Model	15
4.1.2 Comanda	16
4.1.3 Compra	17
4.1.4 Endereço	17
4.1.5 Fornecedor	18
4.1.6 Funcionário	18
4.1.7 Garçom	19
4.1.8 Gerente	20
4.1.9 Ingrediente	20
4.1.10 Item	20
4.1.11 Mesa	21
4.1.12 Pedido	21
4.1.13 Produto	22
4.2 Controles	23
4.2.1 Controle de Atendimento	23
4.2.2 Controle de Compra	24
4.2.3 Controle de Fornecedor	24
4.2.4 Controle de Funcionário	24
4.2.5 Controle de Impressão	24
4.2.6 Controle de Item	25

4.2.7	Controle de Produto	25
4.3	Fronteiras	25
4.3.1	Main Activity	25
4.3.2	Menu Principal	26
4.3.3	Menu Fornecedores	27
4.3.4	Menu Funcionários	27
4.3.5	Menu Garçom	28
4.3.6	Menu Itens	28
4.3.7	Menu Produtos	29
4.3.8	Menu Relatório	30
4.3.9	Busca Fornecedor	30
4.3.10	Busca Funcionário	30
4.3.11	Busca Item	30
4.3.12	Busca Produto	30
4.3.13	Cadastro Endereço	30
4.3.14	Cadastro Fornecedor	31
4.3.15	Cadastro Funcionário	31
4.3.16	Cadastro Item	31
4.3.17	Cadastro Produto	31
4.3.18	Criação Comanda	31
4.3.19	Exibir Pedido	32
4.3.20	Selecionar Ingredientes	32
4.3.21	Tela Garçom	32
4.3.22	Tela Relatórios	33
4.4	Adapters	33
4.4.1	Fornecedor Adapter	33
4.4.2	Funcionario Adapter	34
4.4.3	Item Adapter	35
4.4.4	Produto Adapter	35
4.5	Helpers	36
4.5.1	Server Request	36
4.5.2	Request Callback	36
4.5.3	Utilitários	37
4.6	Banco de Dados	39
5	Diagramas	41
5.1	Diagrama de Classes	41
5.2	Diagrama de Casos de Uso	41

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

3 Casos de Uso

Nome: Cadastrar Funcionário.

Descrição: O Gerente cadastra um funcionário no sistema.

Identificador: CDU1.

Importância: 2.

Autor Primário: Gerente.

Fluxo Principal:

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

Nome: Excluir Pedido.

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

Identificador: CDU2.

Importância: 2.

Autor Primário: Garçom.

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

Fluxo Principal:

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

Nome: Alterar Cadastro de Funcionário.

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

Identificador: CDU3.

Importância: 2.

Ator Primário: Gerente.

Fluxo Principal:

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

Nome: Gerar relatório de itens em falta.

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

Identificador: CDU4.

Importância: 1.

Ator Primário: Gerente.

Fluxo Principal:

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

Nome: Alertar sobre itens abaixo do limite.

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

Identificador: CDU5.

Importância: 1.

Autor Primário: Gerente.

Fluxo Principal:

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

4 Código do Projeto

4.1 Modelos

4.1.1 Model

```
1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4 import android.util.Log;
5
6 import com.google.gson.Gson;
7 import com.google.gson.reflect.TypeToken;
8
9 import org.json.JSONException;
10 import org.json.JSONObject;
11
12 import java.lang.reflect.ParameterizedType;
13 import java.lang.reflect.Type;
14 import java.util.ArrayList;
15
16 import ds2.equipe1.restaurante.helpers.RequestCallback;
17 import ds2.equipe1.restaurante.helpers.ServerRequest;
18 import ds2.equipe1.restaurante.helpers.Utils;
19
20 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
47                     super.execute(json);
48                 }
49             });
50     }
51
52     public void delete(){
53         new ServerRequest(context).sendRequest(getControllerName(), ServerRequest.Action.DELETE, new Gson().toJson(this),
54             null);
55     }
56
57     public static <T extends Model> void find(Context context, Class<T> klass, final Type typeArray, final
58     RequestCallback<T> callback, int id){
59         find(context, klass, typeArray, callback, id + "");
60     }
61
62     public static <T extends Model> void find(final Context context, Class<T> klass, final Type typeArray, final
63     RequestCallback<T> callback, String where){
64         ServerRequest serverRequest = new ServerRequest(context);
65         String controller = ((Class<T>) ((ParameterizedType) klass.getGenericSuperclass()).getActualTypeArguments()[0])
66             .getSimpleName();
67         serverRequest.sendRequest(controller, ServerRequest.Action.FIND, where, new RequestCallback<JSONObject>() {
68             @Override
69             public void execute(JSONObject json) throws Exception {
70                 Log.w(Utils.TAG, "Trying to cast to " + typeArray.toString());
71
72                 ArrayList<T> lista = new Gson().fromJson(json.getJSONArray("data").toString(), typeArray);
73                 for (T item : lista){
74                     item.setContext(context);
75                 }
76                 if (callback != null){
77                     if (lista.size() > 0) {
78                         callback.execute(lista.get(0));
79                     }
80                     callback.execute(lista);
81                 }
82             }
83         });
84     }
85
86     public void setId(Integer id){
87         this.id = id;
88     }
89
90     public Integer getId(){
91         return id;
92     }
93
94     public void setContext(Context context){
95         this.context = context;
96     }
97
98     public Context getContext(){
99         return context;
100    }
101 }
```

```

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

```

Código 1: Model.java

4.1.2 Comanda

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6 import java.util.Date;
7
8 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 2: Comanda.java

4.1.3 Compra

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

Código 3: Compra.java

4.1.4 Endereço

```
1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 public class Endereco extends Model<Endereco> {
6     private String logradouro;
7     private String rua;
8     private int numero;
9     private String bairro;
10    private String cidade;
11    private String estado;
12    private String cep;
13
14    public Endereco(Context context, String logradouro, String rua, int numero, String bairro, String cidade, String
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() {
```

```

46     return cep;
47 }
48
49 public String getEstado() {
50     return estado;
51 }
52
53 public void setEstado(String estado) {
54     this.estado = estado;
55 }
56
}

```

Código 4: Endereco.java

4.1.5 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
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 5: Fornecedor.java

4.1.6 Funcionário

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4 import android.content.Intent;
5
6 public class Funcionario extends Model<Funcionario> {
7     private String nome;
8     private Integer id_endereco;
9     private String telefone;
10    private String cpf;
11    private String nome_de_usuario;
12    private Integer tipo;
13
14    public Funcionario(Context context){
15        super(context);
16    }
17
18    public Funcionario(Context context, String nome, Integer id_endereco, String telefone, String cpf, String
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    public void setTelefone (String telefone){
56        this.telefone = telefone;
57    }
58
59    public void setCpf(String cnpj){
60        this.cpf = cnpj;
61    }
62
63    public void setIdEndereco(Integer id_endereco) {
64        this.id_endereco = id_endereco;
65    }
66
67    public void setNome_de_usuario (String nome_de_usuario) { this.nome_de_usuario = nome_de_usuario; }
68
69    public void setTipo (Integer tipo) {this.tipo = tipo; }
}

```

Código 6: Funcionario.java

4.1.7 Garçom

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 public class Garcom extends Funcionario {
6     public Garcom(Context context, String nome, 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     }
}

```

Código 7: Garcom.java

4.1.8 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     }
}
```

Código 8: Gerente.java

4.1.9 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 9: Ingrediente.java

4.1.10 Item

```
1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 import ds2.equipe1.restaurante.controles.ControleDeAtendimento;
6
7 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 10: Item.java

4.1.11 Mesa

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 public class Mesa extends Model<Mesa> {
8     private int numero;
9     private ArrayList<Comanda> comandas = new ArrayList< Comanda >();
10
11    public Mesa(Context context, int numero){
12        super(context);
13        //inserir chamada do SQL para carregar comandas ativas
14        this.numero = numero;
15    }
16
17    public void addComanda(Comanda nova){
18        comandas.add(nova);
19    }
20
21    public ArrayList< Comanda > getComandas(){
22        return comandas;
23    }
24
25    public ArrayList < Comanda > getComandasAtivas(){
26        ArrayList < Comanda > ativas = new ArrayList < Comanda >();
27        for (int i = 0; i < comandas.size(); i++) {
28            Comanda atual = comandas.get(i);
29            if(atual.estaaAtiva()){
30                ativas.add(atual);
31            }
32        }
33        return ativas;
34    }
35
36    public Comanda getComanda(int indice){
37        return comandas.get(indice);
38    }
39 }
```

Código 11: Mesa.java

4.1.12 Pedido

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 public class Pedido extends Model<Pedido> {
8     int quantidadeDeProdutos;
```

```

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

Código 12: Pedido.java

4.1.13 Produto

```

1 package ds2.equipe1.restaurante.modelos;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 public class Produto extends Model<Produto> {
8     private String nome;
9     private float preco;
10    private ArrayList<Ingrediente> ingredientes;
11
12    public Produto(Context context, String nome, float preco, ArrayList<Ingrediente> ingredientes) {
13        super(context);
14        this.nome = nome;
15        this.preco = preco;
16        this.ingredientes = ingredientes;
17    }
18
19    /*public static ArrayList<Produto> carregarProdutos() {
20        ArrayList<Produto> listaDeProdutos = new ArrayList<Produto>();
21
22        ArrayList<Produto> lista = new ArrayList<Produto>;
23
24        //conexao com o banco estabelecida
25        for (int i = 0; i < lista.count(); i++) {
26            listaDeProdutos.add(i, new Produto(lista.nome, lista.preco, lista.id, ...));
27        }
28
29        return listaDeProdutos; //(?)
30    }*/
31
32    public boolean validarProduto() {
33        /*for (int i = 0; i < ingredientes.size(); i++) {
34            if (ingredientes.get(i).getQuantidade() > ControleDeItem.consultarItem(ingredientes.get(i).getItem().getNome(), new RequestCallback<Item>() {
35                @Override
36                public void execute(ArrayList<Item> itens) {
37                    BuscaItem.this.itens.clear();
38                    BuscaItem.this.itens.addAll(itens);
39                    adapter.notifyDataSetChanged();
40                    super.execute(itens);
41                }
42            }).getQuantidade()) {
43                return false;
44            }
45        }*/
46        return true;
47    }
48 }
```

```

49 //Deve ser chamado quando o garcon conclui o pedido.
50 public void alertarSobreItensAbaixoDoLimite(){
51     for (Ingrediente ingrediente : getIngredientes()){
52         Item item = ingrediente.getItem();
53         item.verificarItemAbaixoDoLimite(ingrediente.getQuantidade());
54     }
55 }
56
57 public String getNome(){
58     return nome;
59 }
60
61 public float getPreco(){
62     return preco;
63 }
64
65 public ArrayList <Ingrediente> getIngredientes(){
66     return ingredientes;
67 }
68
69 public void setPreco(float preco){
70     this.preco = preco;
71 }
72 }
```

Código 13: Produto.java

4.2 Controles

4.2.1 Controle de Atendimento

```

1 package ds2.equipe1.restaurante.controles;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 import ds2.equipe1.restaurante.modelos.Comanda;
8 import ds2.equipe1.restaurante.modelos.Mesa;
9 import ds2.equipe1.restaurante.modelos.Pedido;
10 import ds2.equipe1.restaurante.modelos.Produto;
11
12 /**
13 * Created by Th on 24/03/2016.
14 */
15 public class ControleDeAtendimento {
16     private Context context;
17     private ControleDeImpressao controleDeImpressao;
18
19     private ArrayList<Mesa> mesas = new ArrayList<Mesa>();
20
21     public ControleDeAtendimento(Context context){
22         this.context = context;
23         this.controleDeImpressao = new ControleDeImpressao(context);
24
25         //inserir comando sql pra saber numero de mesas
26         int numeroDeMesas = 20; //puxar numero do sql
27         for(int i = 1; i <= numeroDeMesas; i++) {
28             mesas.add(new Mesa(context, i));
29         }
30     }
31
32     public void criarComanda(Mesa mesa, Pedido pedido, String nome){
33         mesa.addComanda(new Comanda(context, nome, pedido));
34     }
35
36     public void encerrarComanda(Comanda comanda){
37         comanda.desativar();
38     }
39
40     public void imprimirPedido(Pedido pedido){
41         ArrayList<Produto> listaDeProdutos = pedido.getProdutos();
42         for(int i = 0; i < listaDeProdutos.size(); i++){
43             controleDeImpressao.imprimir(listaDeProdutos.get(i).getNome());
44         }
45     }
46
47     public void imprimirConta(Comanda comanda){
48         String saida = "";
49         ArrayList < Pedido > listaDePedidos = comanda.getPedidos();
50         for(int i = 0; i < listaDePedidos.size(); i++) {
51             ArrayList < Produto > listaDeProdutos = listaDePedidos.get(i).getProdutos();
52             for(int j = 0; j < listaDeProdutos.size(); j++) {
53                 saida += listaDeProdutos.get(j).getNome() + " " + listaDeProdutos.get(j).getPreco() + "\n";
54             }
55         }
56     }
57 }
```

```

56     float total = comanda.getCustoTotal();
57     saida += "Total de Pedidos" + total + "\n10% = " + 0.1*total + "\nTotal " + 1.1*total;
58     controleDeImpressao.imprimir(saida);
59 }
60
61 public ArrayList < Produto > consultarPedido(Pedido pedido){
62     return pedido.getProdutos();
63 }
64
65 public void incluirPedido(Comanda comanda, Pedido pedido){
66     comanda.addPedido(pedido);
67 }
68
69 public void entregarPedido(Pedido pedido){
70     pedido.setEntregue();
71 }
72
73 public void cancelarPedido(Comanda comanda, Pedido pedido) {
74     comanda.removerPedido(pedido);
75 }
76
77 public ArrayList < Produto > consultarCardapio() {
78     ArrayList < Produto > cardapio = new ArrayList < Produto >();
79     //Aqui so com consulta ao DB ne? pq precisa pegar a lista de produtos.
80     //fica pra outra hora beijo no coracao!
81     return cardapio;
82 }
83
84 public ArrayList < Comanda > relatorioComandasAtivas(){
85     ArrayList < Comanda > ativas = new ArrayList < Comanda >();
86     //preciso de consulta ao banco neste ponto
87     return ativas;
88 }
89
90
91 public ArrayList < Comanda > relatorioComandasEncerradas(){
92     ArrayList < Comanda > encerradas = new ArrayList < Comanda >();
93     //preciso de consulta ao banco neste ponto
94     return encerradas;
95 }
96
97 public void relatorioArrecadacaoBruta(){
98     //preciso de consulta ao banco neste ponto
99 }
100
101
102 }

```

Código 14: ControleDeAtendimento.java

4.2.2 Controle de Compra

4.2.3 Controle de Fornecedor

4.2.4 Controle de Funcionário

4.2.5 Controle de Impressão

```

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

```

Código 15: ControleDeImpressao.java

4.2.6 Controle de Item

4.2.7 Controle de Produto

```
1 package ds2.equipe1.restaurante.controles;
2
3 import android.content.Context;
4
5 import java.util.ArrayList;
6
7 import ds2.equipe1.restaurante.modelos.Ingrediente;
8 import ds2.equipe1.restaurante.modelos.Item;
9 import ds2.equipe1.restaurante.modelos.Produto;
10 import java.sql.*;
11
12 /**
13 * Created by Th on 24/03/2016.
14 */
15 public class ControleDeProduto {
16
17     private ArrayList <Produto> produtos;
18     private Context context;
19
20     public ControleDeProduto(Context context){
21         this.context = context;
22     }
23
24
25     public void carregarProdutosDoBanco(){
26         produtos = new ArrayList <Produto>();
27         int tamanhoDoCardapio = 5; //Ler o tamanho do banco
28
29         for (int i = 0; i < tamanhoDoCardapio; i++) {
30             String nome = "teste"; //Ler o nome do banco
31             //produtos.add(new Produto(nome));
32         }
33     }
34
35     public void cadastrarProduto(String nome, float preco, ArrayList <Ingrediente> ingredientes){
36         produtos.add(new Produto(context, nome, preco, ingredientes));
37     }
38
39     public void excluirProduto(String nome){
40         for (int i = 0; i < produtos.size(); i++) {
41             if (produtos.get(i).getNome() == nome) {
42                 //Remover do banco
43                 produtos.remove(i); //ver se e este mesmo o metodo de remocao
44             }
45         }
46     }
47
48     public void alterarPrecoDeProduto(String nome, float novoPreco){
49         for (int i = 0; i < produtos.size(); i++) {
50             if (produtos.get(i).getNome() == nome) {
51                 //Alterar preco no banco
52                 produtos.get(i).setPreco(novoPreco);
53             }
54         }
55     }
56
57     public Produto consultarProduto(String nome){
58         for (int i = 0; i < produtos.size(); i++) {
59             if (produtos.get(i).getNome() == nome) {
60                 return produtos.get(i);
61             }
62         }
63         return null;
64     }
65
66     public void relatorioProdutos(){
67         //Gerar pdf?
68     }
69
70     public void relatorioItensPorProduto(){
71         //Gerar pdf?
72     }
73 }
74 }
```

Código 16: ControleDeProduto.java

4.3 Fronteiras

4.3.1 Main Activity

```

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

Código 17: MainActivity.java

4.3.2 Menu Principal

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.Menu;
7 import android.view.MenuItem;
8 import android.view.View;
9
10 import ds2.equipe1.restaurante.controles.ControleDeFornecedor;
11 import ds2.equipe1.restaurante.controles.ControleDeImpressao;
12 import ds2.equipe1.restaurante.helpers.Utils;
13 import ds2.equipe1.restaurante.modelos.Fornecedor;
14 import ds2.equipe1.restaurante.modelos.Funcionario;
15
16 public class MenuPrincipal extends AppCompatActivity {
17
18     @Override
19     protected void onCreate(Bundle savedInstanceState) {
20         super.onCreate(savedInstanceState);
21         setContentView(R.layout.activity_menu_principal);
22
23         findViewById(R.id.menu_fornecedores).setOnClickListener(new View.OnClickListener() {
24             @Override
25             public void onClick(View v) {
26                 open(MenuFornecedores.class);
27             }
28         });
29
30         findViewById(R.id.menu_garcom).setOnClickListener(new View.OnClickListener() {
31             @Override
32             public void onClick(View v) {
33                 open(TelaGarcom.class);
34             }
35         });
36
37         findViewById(R.id.menu_itens).setOnClickListener(new View.OnClickListener() {
38             @Override
39             public void onClick(View v) {
40                 open(MenuItens.class);
41             }
42         });
43
44         findViewById(R.id.menu_produto).setOnClickListener(new View.OnClickListener() {
45             @Override
46             public void onClick(View v) {
47                 open(MenuProdutos.class);
48             }
49         });
50
51         findViewById(R.id.menu_relatorios).setOnClickListener(new View.OnClickListener() {
52             @Override
53             public void onClick(View v) {
54                 open(MenuRelatorio.class);
55             }
56         });
57
58         findViewById(R.id.menu_funcionarios).setOnClickListener(new View.OnClickListener() {
59             @Override
60             public void onClick(View v) {
61                 open(MenuFuncionarios.class);
62             }
63         });
64
65
66     @Override
67     public boolean onCreateOptionsMenu(Menu menu) {
```

```

68     // Inflate the menu; this adds items to the action bar if it is present.
69     getMenuInflater().inflate(R.menu.menu_principal, menu);
70     return true;
71 }
72
73 @Override
74 public boolean onOptionsItemSelected(MenuItem item) {
75     // Handle action bar item clicks here. The action bar will
76     // automatically handle clicks on the Home/Up button, so long
77     // as you specify a parent activity in AndroidManifest.xml.
78     int id = item.getItemId();
79
80     if (id == R.id.configurar_host) {
81         final Utils utils = new Utils(this);
82         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     startActivity(new Intent(this, activity));
96 }
97 }
```

Código 18: MenuPrincipal.java

4.3.3 Menu Fornecedores

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7
8 public class MenuFornecedores extends AppCompatActivity {
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_menu_fornecedores);
14
15
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 19: MenuFornecedores.java

4.3.4 Menu Funcionários

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7
```

```

8 public class MenuFuncionarios extends AppCompatActivity {
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_menu_funcionarios);
14
15        findViewById(R.id.menu_cadastrar_funcionario).setOnClickListener(new View.OnClickListener() {
16            @Override
17            public void onClick(View v) {
18                open(CadastroFuncionario.class);
19            }
20        });
21
22        findViewById(R.id.menu_buscar_funcionario).setOnClickListener(new View.OnClickListener() {
23            @Override
24            public void onClick(View v) {
25                open(BuscaFuncionario.class);
26            }
27        });
28
29        findViewById(R.id.menu_alterar_funcionario).setOnClickListener(new View.OnClickListener() {
30            @Override
31            public void onClick(View v) {
32                open(CadastroFuncionario.class);
33            }
34        });
35
36        public void open(Class activity){
37            startActivity(new Intent(this, activity));
38        }
39    }
40}

```

Código 20: MenuFuncionarios.java

4.3.5 Menu Garçom

```

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

```

Código 21: MenuGarcom.java

4.3.6 Menu Itens

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7
8 public class 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 22: MenuItens.java

4.3.7 Menu Produtos

```

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

```

Código 23: MenuProdutos.java

4.3.8 Menu Relatório

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

Código 24: MenuRelatorio.java

4.3.9 Busca Fornecedor

4.3.10 Busca Funcionário

4.3.11 Busca Item

4.3.12 Busca Produto

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

Código 25: BuscaProduto.java

4.3.13 Cadastro Endereço

```
1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7 import android.widget.Button;
8 import android.widget.EditText;
9
10 import org.json.JSONObject;
11
12 import ds2.equipe1.restaurante.helpers.RequestCallback;
13 import ds2.equipe1.restaurante.modelos.Endereco;
14 import ds2.equipe1.restaurante.modelos.Model;
15
16 public class CadastroEndereco extends AppCompatActivity {
17
18     private EditText edtCEP, edtLogradouro, edtRua, edtBairro, edtCidade, edtEstado, edtNumero;
19     private Button btnCadastrar;
20
21     @Override
22     protected void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.activity_cadastro_endereco);
25
26         init();
27     }
28 }
```

```

29     private void init() {
30         btnCadastrar = (Button) findViewById(R.id.btnCadastrar);
31         edtRua = (EditText) findViewById(R.id.edtRua);
32         edtCEP = (EditText) findViewById(R.id.edtCEP);
33         edtLogradouro = (EditText) findViewById(R.id.edtLogradouro);
34         edtBairro = (EditText) findViewById(R.id.edtBairro);
35         edtCidade = (EditText) findViewById(R.id.edtCidade);
36         edtEstado = (EditText) findViewById(R.id.edtEstado);
37         edtNumero = (EditText) findViewById(R.id.edtNumero);
38
39         btnCadastrar.setOnClickListener(new View.OnClickListener() {
40             @Override
41             public void onClick(View v) {
42
43                 final Endereco endereco = new Endereco(CadastroEndereco.this, edtLogradouro.getText().toString(),
44                     edtRua.getText().toString(), Integer.parseInt(edtNumero.getText().toString()), edtBairro.getText().toString(),
45                     edtCidade.getText().toString(), edtEstado.getText().toString(), edtCEP.getText().toString());
46
47                 endereco.save(new RequestCallback<Model>() {
48                     @Override
49                     public void execute(Model model) throws Exception {
50                         super.execute(model);
51
52                         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 26: CadastroEndereco.java

4.3.14 Cadastro Fornecedor

4.3.15 Cadastro Funcionário

4.3.16 Cadastro Item

4.3.17 Cadastro Produto

4.3.18 Criação Comanda

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Context;
4 import android.content.Intent;
5 import android.support.v7.app.AppCompatActivity;
6 import android.os.Bundle;
7 import android.view.Menu;
8 import android.view.MenuItem;
9 import android.view.View;
10
11 import ds2.equipe1.restaurante.controles.ControleDeAtendimento;
12 import ds2.equipe1.restaurante.modelos.Comanda;
13
14 public class CriacaoComanda extends AppCompatActivity {
15
16     private ControleDeAtendimento controleDeAtendimento;
17     Comanda comanda;
18     private Context context;
19
20
21     @Override
22     protected void onCreate(Bundle savedInstanceState) {
23         super.onCreate(savedInstanceState);
24         setContentView(R.layout.activity_criacao_comanda);
25         controleDeAtendimento = new ControleDeAtendimento(this);
26
27         findViewById(R.id.btnAdicionarPedido).setOnClickListener(new View.OnClickListener() {
28             @Override
29             public void onClick(View v) {
30                 startActivity(new Intent(CriacaoComanda.this, BuscaProduto.class));
31             }
32         });
33
34         findViewById(R.id.btnEncerrarComanda).setOnClickListener(new View.OnClickListener() {
35             @Override
36             public void onClick(View v) {

```

```

37         //encerrar comanda
38         //imprimir comanda
39     });
40 }
41
42
43 @Override
44 public boolean onCreateOptionsMenu(Menu menu) {
45     // Inflate the menu; this adds items to the action bar if it is present.
46     getMenuInflater().inflate(R.menu.comandas, menu);
47     return true;
48 }
49
50 @Override
51 public boolean onOptionsItemSelected(MenuItem item) {
52     // Handle action bar item clicks here. The action bar will
53     // automatically handle clicks on the Home/Up button, so long
54     // as you specify a parent activity in AndroidManifest.xml.
55     int id = item.getItemId();
56
57     if (id == android.R.id.home) {
58         onBackPressed();
59         return true;
60     }
61
62     return super.onOptionsItemSelected(item);
63 }
64
65 public void onItemClick(View v){
66     startActivity(new Intent(this, ExibirPedido.class));
67 }
68 }
```

Código 27: CriacaoComanda.java

4.3.19 Exibir Pedido

```

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

Código 28: ExibirPedido.java

4.3.20 Seleccionar Ingredientes

```

1 package ds2.equipe1.restaurante;
2
3 import android.support.v7.app.AppCompatActivity;
4
5 /**
6  * Created by Th on 18/04/2016.
7  */
8 public class SeleccionarIngredientes extends AppCompatActivity {
9
10 }
```

Código 29: SeleccionarIngredientes.java

4.3.21 Tela Garçom

```

1 package ds2.equipe1.restaurante;
2
3 import android.content.Intent;
4 import android.support.v7.app.AppCompatActivity;
5 import android.os.Bundle;
6 import android.view.View;
7
```

```

8  public class TelaGarcom extends AppCompatActivity {
9
10    @Override
11    protected void onCreate(Bundle savedInstanceState) {
12        super.onCreate(savedInstanceState);
13        setContentView(R.layout.activity_tela_garcom);
14
15        findViewById(R.id.menu_criar_comanda).setOnClickListener(new View.OnClickListener() {
16            @Override
17            public void onClick(View v) {
18                open(CriacaoComanda.class);
19            }
20        });
21
22        findViewById(R.id.menu_consulta_cardapio).setOnClickListener(new View.OnClickListener() {
23            @Override
24            public void onClick(View v) {
25                open(BuscaProduto.class);
26            }
27        });
28    }
29
30    public void open(Class activity){
31        startActivity(new Intent(this, activity));
32    }
33 }
```

Código 30: TelaGarcom.java

4.3.22 Tela Relatórios

4.4 Adapters

4.4.1 Fornecedor Adapter

```

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

```

51     edtNome = (TextView) root.findViewById(R.id.tvNome);
52     edtCNPJ = (TextView) root.findViewById(R.id.tvCNPJ);
53     edtTelefone = (TextView) root.findViewById(R.id.tvTelefone);
54
55     Fornecedor fornecedor = fornecedores.get(position);
56
57     edtNome.setText(fornecedor.getNome());
58     edtCNPJ.setText(fornecedor.getCnpj());
59     edtTelefone.setText(fornecedor.getTelefone());
60
61     return root;
62 }
}

```

Código 31: FornecedorAdapter.java

4.4.2 Funcionario Adapter

```

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

```

Código 32: FuncionarioAdapter.java

4.4.3 Item Adapter

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

Código 33: ItemAdapter.java

4.4.4 Produto Adapter

```
1 package ds2.equipe1.restaurante.listas;
2
3 import android.content.Context;
4 import android.view.LayoutInflater;
5 import android.view.View;
6 import android.view.ViewGroup;
7 import android.widget.BaseAdapter;
8 import android.widget.TextView;
9
10 import java.util.ArrayList;
11
12 import ds2.equipe1.restaurante.R;
13 import ds2.equipe1.restaurante.modelo.Produto;
14
15 /**
16 * Created by Fernando on 08/04/2016.
17 */
18 public class ProdutoAdapter extends BaseAdapter {
19     private Context context;
20     private ArrayList<Produto> produtos;
```

```

1 public ProdutoAdapter(Context context, ArrayList<Produto> produtos) {
2     this.context = context;
3     this.produtos = produtos;
4 }
5
6     @Override
7     public int getCount() {
8         return produtos.size();
9     }
10
11     @Override
12     public Produto getItem(int position) {
13         return produtos.get(position);
14     }
15
16     @Override
17     public long getItemId(int position) {
18         return produtos.get(position).getId();
19     }
20
21     @Override
22     public View getView(int position, View root, ViewGroup parent) {
23         if (root == null) {
24             LayoutInflater vi = (LayoutInflater) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
25             root = vi.inflate(R.layout.item_produto, parent, false);
26         }
27
28         TextView edtNome, edtPreco;
29         edtNome = (TextView) root.findViewById(R.id.tvNome);
30         edtPreco = (TextView) root.findViewById(R.id.tvPreco);
31
32         Produto produto = produtos.get(position);
33
34         edtNome.setText(produto.getNome());
35         edtPreco.setText(Float.toString(produto.getPreco()));
36
37         return root;
38     }
39 }

```

Código 34: ProdutoAdapter.java

4.5 Helpers

4.5.1 Server Request

4.5.2 Request Callback

```

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

```

```

37     onFinish();
38 }
39
40 public void onFinish(){
41     if (context != null) {
42         Utils.dismissProgress();
43     }
44 }
45

```

Código 35: RequestCallback.java

4.5.3 Utilitários

```

1 package ds2.equipe1.restaurante.helpers;
2
3 import android.app.AlertDialog;
4 import android.content.Context;
5 import android.content.DialogInterface;
6 import android.content.SharedPreferences;
7 import android.support.v7.app.AlertDialog;
8 import android.view.LayoutInflater;
9 import android.widget.EditText;
10 import android.widget.LinearLayout;
11 import android.widget.Spinner;
12 import android.widget.Toast;
13
14 import ds2.equipe1.restaurante.R;
15
16 /**
17 * Created by Fernando on 28/03/2016.
18 */
19 public class Utils {
20     public static final String TAG = "Restaurante";
21     private Context context;
22     private static ProgressDialog progress;
23
24     public Utils(Context context) {
25         this.context = context;
26     }
27
28     public void deleteData(String key){
29         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
30         SharedPreferences.Editor e = settings.edit();
31         e.remove(key);
32         e.commit();
33     }
34
35     public float getData(String key, float defaultValue) {
36         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
37         return settings.getFloat(key, defaultValue);
38     }
39
40     public String getData(String key, String defaultValue) {
41         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
42         return settings.getString(key, defaultValue);
43     }
44
45     public int getData(String key, int defaultValue) {
46         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
47         return settings.getInt(key, defaultValue);
48     }
49
50     public long getData(String key, long defaultValue) {
51         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
52         return settings.getLong(key, defaultValue);
53     }
54
55     public boolean getData(String key, boolean defaultValue) {
56         SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
57         return settings.getBoolean(key, defaultValue);
58     }
59
60     public void setData(String key, float value) {
61         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
62         settings.putFloat(key, value);
63         settings.commit();
64     }
65
66     public void setData(String key, String value) {
67         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
68         settings.putString(key, value);
69         settings.commit();
70     }
71
72     public void setData(String key, int value) {
73         SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
74         settings.putInt(key, value);
75     }
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
317
318
319
319
320
321
322
323
324
325
326
327
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
387
388
389
389
390
391
392
393
394
395
396
397
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
417
418
419
419
420
421
422
423
424
425
426
427
427
428
429
429
430
431
432
433
434
435
436
437
437
438
439
439
440
441
442
443
444
445
445
446
447
447
448
449
449
450
451
452
453
454
455
456
456
457
457
458
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
717
718
719
719
720
721
722
723
724
725
726
727
727
728
729
729
730
731
732
733
734
735
736
737
737
738
739
739
740
741
742
743
744
745
745
746
747
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
817
818
819
819
820
821
822
823
824
825
826
827
827
828
829
829
830
831
832
833
834
835
836
837
837
838
839
839
840
841
842
843
844
845
846
847
847
848
849
849
850
851
852
853
854
855
856
857
858
858
859
860
861
862
863
864
865
866
867
867
868
869
869
870
871
872
873
874
875
876
877
877
878
879
879
880
881
882
883
884
885
886
887
887
888
889
889
890
891
892
893
894
895
896
897
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
917
918
919
919
920
921
922
923
924
925
926
927
927
928
929
929
930
931
932
933
934
935
936
937
937
938
939
939
940
941
942
943
944
945
946
946
947
948
948
949
949
950
951
952
953
954
955
956
957
957
958
959
959
960
961
962
963
964
965
966
966
967
968
968
969
969
970
971
972
973
974
975
976
977
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1046
1047
1048
1048
1049
1049
1050
1051
1052
1053
1054
1055
1056
1057
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1066
1067
1068
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1076
1077
1078
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1086
1087
1088
1088
1089
1089
1090
1091
1092
1093
1094
1095
1095
1096
1097
1097
1098
1098
1099
1099
1100
1101
1102
1103
1104
1105
1105
1106
1107
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1115
1116
1117
1117
1118
1118
1119
1119
1120
1121
1122
1123
1124
1125
1125
1126
1127
1127
1128
1128
1129
1129
1130
1131
1132
1133
1134
1135
1135
1136
1137
1137
1138
1138
1139
1139
1140
1141
1142
1143
1144
1145
1145
1146
1147
1147
1148
1148
1149
1149
1150
1151
1152
1153
1154
1155
1155
1156
1157
1157
1158
1158
1159
1159
1160
1161
1162
1163
1164
1165
1165
1166
1167
1167
1168
1168
1169
1169
1170
1171
1172
1173
1174
1174
1175
1176
1176
1177
1177
1178
1178
1179
1179
1180
1181
1182
1183
1184
1185
1185
1186
1187
1187
1188
1188
1189
1189
1190
1191
1192
1193
1194
1194
1195
1196
1196
1197
1197
1198
1198
1199
1199
1200
1201
1202
1203
1204
1204
1205
1206
1206
1207
1207
1208
1208
1209
1209
1210
1211
1212
1213
1214
1214
1215
1216
1216
1217
1217
1218
1218
1219
1219
1220
1221
1222
1223
1224
1225
1225
1226
1227
1227
1228
1228
1229
1229
1230
1231
1232
1233
1234
1235
1235
1236
1237
1237
1238
1238
1239
1239
1240
1241
1242
1243
1244
1244
1245
1246
1246
1247
1247
1248
1248
1249
1249
1250
1251
1252
1253
1254
1255
1255
1256
1257
1257
1258
1258
1259
1259
1260
1261
1262
1263
1264
1265
1265
1266
1267
1267
1268
1268
1269
1269
1270
1271
1272
1273
1274
1274
1275
1276
1276
1277
1277
1278
1278
1279
1279
1280
1281
1282
1283
1284
1284
1285
1286
1286
1287
1287
1288
1288
1289
1289
1290
1291
1292
1293
1294
1294
1295
1296
1296
1297
1297
1298
1298
1299
1299
1300
1301
1302
1303
1304
1304
1305
1306
1306
1307
1307
1308
1308
1309
1309
1310
1311
1312
1313
1314
1314
1315
1316
1316
1317
1317
1318
1318
1319
1319
1320
1321
1322
1323
1324
1324
1325
1326
1326
1327
1327
1328
1328
1329
1329
1330
1331
1332
1333
1334
1334
1335
1336
1336
1337
1337
1338
1338
1339
1339
1340
1341
1342
1343
1344
1344
1345
1346
1346
1347
1347
1348
1348
1349
1349
1350
1351
1352
1353
1354
1354
1355
1356
1356
1357
1357
1358
1358
1359
1359
1360
1361
1362
1363
1364
1364
1365
1366
1366
1367
1367
1368
1368
1369
1369
1370
1371
1372
1373
1374
1374
1375
1376
1376
1377
1377
1378
1378
1379
1379
1380
1381
1382
1383
1384
1384
1385
1386
1386
1387
1387
1388
1388
1389
1389
1390
1391
1392
1393
1394
1394
1395
1396
1396
1397
1397
1398
1398
1399
1399
1400
1401
1402
1403
1404
1404
1405
1406
1406
1407
1407
1408
1408
1409
1409
1410
1411
1412
1413
1414
1414
1415
1416
1416
1417
1417
1418
1418
1419
1419
1420
1421
1422
1423
1424
1424
1425
1426
1426
1427
1427
1428
1428
1429
1429
1430
1431
1432
1433
1434
1434
1435
1436
1436
1437
1437
1438
1438
1439
1439
1440
1441
1442
1443
1444
1444
1445
1446
1446
1447
1447
1448
1448
1449
1449
1450
1451
1452
1453
1454
1454
1455
1456
1456
1457
1457
1458
1458
1459
1459
1460
1461
1462
1463
1464
1464
1465
1466
1466
1467
1467
1468
1468
1469
1469
1470
1471
1472
1473
1474
1474
1475
1476
1476
1477
1477
1478
1478
1479
1479
1480
1481
1482
1483
1484
1484
1485
1486
1486
1487
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1494
1495
1496
1496
1497
1497
1498
1498
1499
1499
1500
1501
1502
1503
1504
1504
1505
1506
1506
1507
1507
1508
1508
1509
1509
1510
1511
1512
1513
1514
1514
1515
1516
1516
1517
1517
1518
1518
1519
1519
1520
1521
1522
1523
1524
1524
1525
1526
1526
1527
1527
1528
1528
1529
1529
1530
1531
1532
1533
1534
1534
1535
1536
1536
1537
1537
1538
1538
1539
1539
1540
1541
1542
1543
1544
1544
1545
1546
1546
1547
1547
1548
1548
1549
1549
1550
1551
1552
1553
1554
1554
1555
1556
1556
1557
1557
1558
1558
1559
1559
1560
1561
1562
1563
1564
1564
1565
1566
1566
1567
1567
1568
1568
1569
1569
1570
1571
1572
1573
1574
1574
1575
1576
1576
1577
1577
1578
1578
1579
1579
1580
1581
1582
1583
1584
1584
1585
1586
1586
1587
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1594
1595
1596
1596
1597
1597
1598
1598
1599
1599
1600
1601
1602
1603
1604
1604
1605
1606
1606
1607
1607
1608
1608
1609
1609
1610
1611
1612
1613
1614
1614
1615
1616
1616
1617
1617
1618
1618
1619
1619
1620
1621
1622
1623
1624
1624
1625
1626
1626
1627
1627
1628
1628
1629
1629
1630
1631
1632
1633
1634
1634
1635
1636
1636
1637
1637
1638
1638
1639
1639
1640
1641
1642
1643
1644
1644
1645
1646
1646
1647
1647
1648
1648
1649
1649
1650
1651
1652
1653
1654
1654
1655
1656
1656
1657
1657
1658
1658
1659
1659
1660
1661
1662
1663
1664
1664
1665
1666
1666
1667
1667
1668
1668
1669
1669
1670
1671
1672
1673
1674
1674
1675
1676
1676
1677
1677
1678
1678
1679
1679
1680
1681
1682
1683
1684
1684
1685
1686
1686
1687
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1694
1695
1696
1696
1697
1697
1698
1698
1699
1699
1700
1701
1702
1703
1704
1704
1705
1706
1706
1707
1707
1708
1708
1709
1709
1710
1711
1712
1713
1714
1714
1715
1716
1716
1717
1717
1718
1718
1719
1719
1720
1721
1722
1723
1724
1724
1725
1726
1726
1727
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1734
1735
1736
1736
1737
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1744
1745
1746
1746
1747
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1754
1755
1756
1756
1757
1757
1758
1758
1759
1759
1760
1761
1762
1763
1764
1764
1765
1766
1766
1767
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1774
1775
1776
1776
1777
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1784
1785
1786
1786
1787
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1794
1795
1796
1796
1797
1797
1798
1798
1799
1799
1800
1801
1802
1803
1804
1804
1805
1806
1806
1807
1807
1808
1808
1809
1809
1810
1811
1812
1813
1814
1814
1815
1816
1816
1817
1817
1818
1818
1819
1819
1820
1821
1822
1823
1824
1824
1825
1826
1826
1827
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1834
1835
1836
1836
1837
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1844
1845
1846
1846
1847
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1854
1855
185
```

```

75     settings.commit();
76 }
77
78 public void setData(String key, long value) {
79     SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
80     settings.putLong(key, value);
81     settings.commit();
82 }
83
84 public void setData(String key, boolean value) {
85     SharedPreferences.Editor settings = context.getSharedPreferences("EsporteNet", 0).edit();
86     settings.putBoolean(key, value);
87     settings.commit();
88 }
89
90 public void clearData(){
91     SharedPreferences settings = context.getSharedPreferences("EsporteNet", 0);
92     SharedPreferences.Editor editor = settings.edit();
93     editor.clear();
94     editor.commit();
95 }
96
97 public static void launchProgress(Context context){
98     launchProgress(context, null, "Carregando...");
99 }
100
101 public static void launchProgress(Context context, String title, String text) {
102     launchProgress(context, title, text, null);
103 }
104
105 public static void launchProgress(Context context, String title, String text, DialogInterface.OnDismissListener
func) {
106     progress = ProgressDialog.show(context, title, text, true);
107     progress.setCancelable(true);
108     if (func != null) progress.setOnDismissListener(func);
109 }
110
111 public static void dismissProgress() {
112     if (progress != null) {
113         progress.dismiss();
114     }
115 }
116
117 public void toast(String text){
118     Toast.makeText(context, text, Toast.LENGTH_LONG).show();
119 }
120
121 public void inputDialog(String title, String defValue, String hint, final DialogCallback callback){
122     AlertDialog.Builder builder = new AlertDialog.Builder(context);
123     builder.setTitle(title);
124
125     // Set up the input
126     LayoutInflator li = (LayoutInflator) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
127     LinearLayout linearLayout = (LinearLayout) li.inflate(R.layout.input_dialog, null);
128     final EditText edtInput = (EditText) linearLayout.findViewById(R.id.edtInput);
129     edtInput.setHint(hint);
130     edtInput.setText(defValue);
131     builder.setView(linearLayout);
132
133     // Set up the buttons
134     builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
135         @Override
136         public void onClick(DialogInterface dialog, int which) {
137             if (callback != null) {
138                 callback.execute(edtInput.getText().toString());
139             }
140         }
141     });
142     builder.setNegativeButton("Cancelar", null);
143
144     builder.show();
145 }
146
147 public void selectPopup(String title, final DialogCallback callback){
148     AlertDialog.Builder builder = new AlertDialog.Builder(context);
149     builder.setTitle(title);
150
151     // Set up the input
152     LayoutInflator li = (LayoutInflator) context.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
153     LinearLayout linearLayout = (LinearLayout) li.inflate(R.layout.select_popup, null);
154
155     final Spinner spinnerItens = (Spinner) linearLayout.findViewById(R.id.spinnerItens);
156     final EditText edtInput = (EditText) linearLayout.findViewById(R.id.edtInput);
157     builder.setView(linearLayout);
158
159     // Set up the buttons
160     builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
161         @Override
162         public void onClick(DialogInterface dialog, int which) {
163             if (callback != null) {
164                 callback.execute(edtInput.getText().toString());
165             }
166         }
167     });

```

```

168     builder.setNegativeButton("Cancelar", null);
169
170     builder.show();
171 }
172
173     public interface DialogCallback {
174         public void execute(String text);
175     }
176 }
```

Código 36: Utils.java

4.6 Banco de Dados

```

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

```

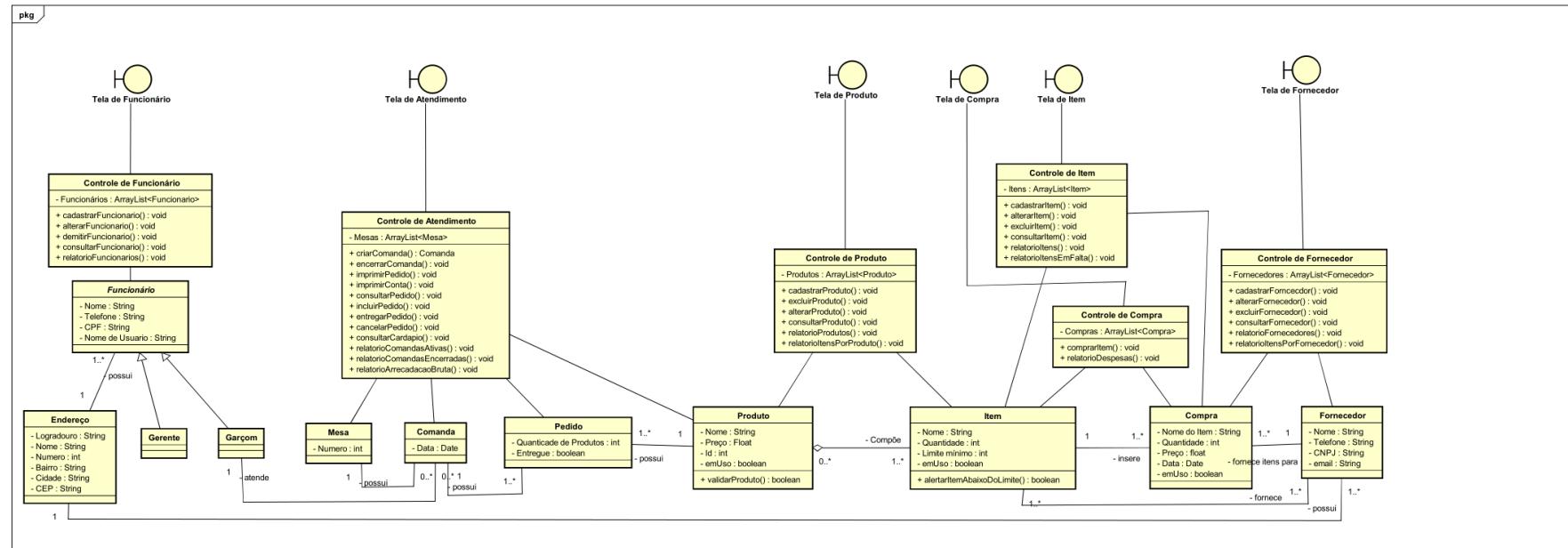
75     CONSTRAINT pk_item PRIMARY KEY (id)
76 );
77
78 CREATE TABLE ingrediente (
79     id_item integer,
80     id_produto integer,
81     quantidade integer,
82     CONSTRAINT pk_item_produto PRIMARY KEY (id_item, id_produto),
83     CONSTRAINT fk_item FOREIGN KEY (id_item) REFERENCES item (id),
84     CONSTRAINT fk_produto FOREIGN KEY (id_produto) REFERENCES produto (id)
85 );
86
87 CREATE TABLE fornecedor (
88     id serial,
89     nome varchar(128),
90     telefone varchar(20),
91     cnpj varchar(25),
92     email varchar(128),
93     id_endereco integer,
94     CONSTRAINT pk_fornecedor PRIMARY KEY (id),
95     CONSTRAINT fk_id_endereco FOREIGN KEY (id_endereco) REFERENCES endereco (id),
96     CONSTRAINT unique_cnpj UNIQUE (cnpj)
97 );
98
99 CREATE TABLE compra (
100    id serial,
101    id_item integer,
102    id_fornecedor integer,
103    quantidade integer,
104    preco float,
105    data timestamp,
106    CONSTRAINT pk_compra PRIMARY KEY (id),
107    CONSTRAINT fk_compra_item FOREIGN KEY (id_item) REFERENCES item (id),
108    CONSTRAINT fk_compra_fornecedor FOREIGN KEY (id_fornecedor) REFERENCES fornecedor (id)
109 );

```

Código 37: CriarBanco.sql

5 Diagramas

5.1 Diagrama de Classes



5.2 Diagrama de Casos de Uso

