CENTRO UNIVERSITÁRIO UNIFACVEST
CURSO DE CIÊNCIA DA COMPUTAÇÃO
DEULLAM JUSTI DOS SANTOS

DCOMANDA: SISTEMA DE AUTOMAÇÃO NO SETOR DE ATENDIMENTO EM RESTAURANTES

deullam@yahoo.com.br
LAGES
2018

DEULLAM JUSTI DOS SANTOS

DCOMANDA: SISTEMA DE AUTOMAÇÃO NO SETOR DE ATENDIMENTO EM RESTAURANTES

Projeto apresentado à Banca Examinadora do Trabalho de Conclusão de Curso II de Ciência da Computação para análise e aprovação.

LAGES

2018

DEULLAM JUSTI DOS SANTOS

DCOMANDA: SISTEMA DE AUTOMAÇÃO NO SETOR DE ATENDIMENTO EM RESTAURANTES

Trabalho de conclusão de curso apresentado ao Centro Universitário UNIFACVEST como parte dos requisitos para a obtenção do grau de Bacharel em Ciência da Computação.

Prof. ESP. Igor Muzeka

Coorientadores:
Prof. Cassandro Devenz
Profa. DRA. Ingrid Solange Sepúlveda Muñoz

Lages, SC ___/___/2018. Nota _____ _____

_____
Coordenador do curso de graduação

LAGES
2018

RESUMO

Este trabalho visa desenvolver um sistema distribuído para gerenciar o setor de atendimento de um restaurante. Informando aos clientes se os ingredientes dos produtos contêm ou não elementos que causam reações alérgicas como glúten e lactose, utilizando os conceitos de sistemas distribuídos e orientação a objeto com tecnologias atuais como java e android. O sistema contará com o uso de três plataformas distintas sendo elas as plataformas desktop, web e a mobile cujo vem crescendo a cada dia. A implantação do sistema poderá proporcionar melhorias no atendimento dos restaurantes, tornando-o mais preciso e diminuindo casos de reações alérgicas, além disso, ainda existem benefícios diferentes como redução de despesas.

Palavras Chave: Sistemas Distribuídos, Automação Comercial, Java, Android.

ABSTRACT

This work aims to develop a distributed system to manage the service sector of a restaurant. Telling customers if product ingredients contain or do not contain elements that cause allergic reactions like gluten and lactose, using the concepts of distributed systems and object orientation with current technologies such as java and android. The system will rely on the use of three distinct platforms: desktop, web and mobile platforms, which are growing every day. The implementation of the system can provide improvements in the service of restaurants, making it more accurate and reducing cases of allergic reactions, and there are still different benefits such as reduced expenses.

Keywords: *Distributed Systems, Commercial Automation, Java, Android.*

RESUMEN

Este trabajo tiene como objetivo desarrollar un sistema distribuido para gestionar el sector de servicios de un restaurante. Informar a los clientes si los ingredientes del producto contienen o no elementos que causan reacciones alérgicas como gluten y lactosa, utilizando los conceptos de sistemas distribuidos y orientación a objetos con las tecnologías actuales, como java y android. El sistema se basará en el uso de tres plataformas distintas: escritorio, web y plataformas móviles, que están creciendo día a día. La implementación del sistema puede proporcionar mejoras en el servicio de los restaurantes, por lo que es más preciso y reduce los casos de reacciones alérgicas, y todavía hay diferentes beneficios, como la reducción de los gastos.

Palabras clave: Sistemas Distribuidos, Automatización Comercial, Java, Android.

LISTA DE SIGLAS

CSS - Cascading Style Sheets

HTML - Hipertext Markup Language

IDE - Integrated Developed Environment

JSP- Java Server Pages

OMG - Object Management Group

OMT - *Object Modeling Technique*

OOSE - *Object-Oriented Software Engineering*

SDK - *Software Development Kit*

UML - Unified Modeling Language

WWW - World wide web

XML - Etensible Markup Language

LISTA DE FIGURAS

Figura 1: Fragmento de código Java 10
Figura 2: Janela da IDE eclipse 11
Figura 3: Cronologia Android 12
Figura 4: Tela da IDE Android Studio 12
Figura 5: Exemplo de código de HTML 14
Figura 6: Exemplo de código CSS 15
Figura 7: Exemplo de código JSP 16
Figura 8: Tela Inicial do Sistema Desktop 19
Figura 9: Tela com informações das mesas. 20
Figura 10: Tela com informações da mesa. 20
Figura 11:Tela de Produtos 21
Figura 12: Tela de Cadastro de Produtos 21
Figura 13: Tela de Cadastro de Ingredientes 22
Figura 14: Tela do Cardápio 22
Figura 15: Tela de informação detalha do produto. 23
Figura 16: Tela de pedido. 23
Figura 17: Tela de formulário para reserva. 24
Figura 18: Tela de cardápio do sistema mobile 25
Figura 19: Tela de informações detalhadas do Produto no app mobile 25
Figura 20: Tela de pedido 26

LISTA DE QUADROS

Quadro 1: Comparativo de custos e benefícios da automação.....4
Quadro 2: Os principais diagramas e conceitos.....6
Quadro 3: Conceitos de orientação a objetos.....7
Quadro 4: Principais componentes do Android.....13
Quadro 5: Cronograma TCC 2.....30

SUMÁRIO

1 INTRODUÇÃO **1**
1.1 JUSTIFICATIVA 1
1.2 IMPORTÂNCIA 1
1.3 OBJETIVO GERAL 2
1.3.1 Objetivos Específicos *2*
2 REVISÃO DE LITERATURA **3**
2.1 AUTOMAÇÃO 3
2.1.1 Objetivos e Benefícios da Automação Comercial *3*
2.2 UML 5
2.2.1 História do UML *5*
2.2.2 Diagramas *5*
2.3 ORIENTAÇÃO A OBJETO 7
2.4 SISTEMAS DISTRIBUÍDOS 8
2.5 JAVA 8
2.5.1 Breve História da Linguagem JAVA *9*
2.5.2 IDE Eclipse *10*
2.6 ANDROID 11
2.6.1 IDE Android *12*
2.6.2 Principais componentes *13*
2.7 PROGRAMAÇÃO WEB 13
2.7.1 World Wide Web *13*
2.7.2 HTML e CSS *14*
2.7.3 JavaScript *15*
2.7.4 JSP *16*
3 METODOLOGIA **17**
3.1 DOCUMENTAÇÃO 17
3.2 NATUREZA DA PESQUISA 17
3.3 TIPO DA PESQUISA 17
3.4 TÉCNICAS DE PESQUISA 17
3.5 COLETA DE DADOS 18
4 PROJETO **19**
4.1 VISÃO GERAL 19
4.2 SISTEMA DESKTOP 19
4.2.1 Módulo do Pedido e Mesas *20*
4.2.2 Módulo de Produtos Ingredientes *21*
4.3 SISTEMA WEB *22*
4.3.1 Cardápio e Pedido *22*
4.3.2 Reservas. *24*
4.4 SISTEMA MOBILE 24
4.5 DIAGRAMAS 26
4.5.1 Diagrama de Caso de Uso *26*
4.5.2 Diagramas de atividade *27*
5 CRONOGRAMA **30**
6 TRABALHOS CORRELATOS **31**
6.1 RESTAURANTE WIRELESS 31
7 RESULTADOS **32**
8 REFERÊNCIAS **33**
APÊNDICE A - CÓDIGO TELA MESAS **35**
APÊNDICE B - CÓDIGO TELA MESA PEDIDO **37**
APÊNDICE C – CÓDIGO DA TELA INDEX **40**
APÊNDICE D – CÓDIGO DO CARDÁPIO **42**
APÊNDICE E – CÓDIGOS DAS CLASSES USADAS EM RESERVAS **44**
APÊNDICE F – PEDIDODAO **46**

1 INTRODUÇÃO

Com os diversos avanços tecnológicos que ocorreram nas últimas décadas e com o surgimento do computador pessoal na década de 80, influenciou de maneira significativa a forma de se produzir bens e serviços, assim aumentando a competividade entre empresas, pois seus clientes querem produtos e serviços cada vez mais baratos e com melhor qualidade (LEME, 2010).

Neste trabalho são apresentados conceitos gerais do desenvolvimento de software e de sua modelagem, com as suas ferramentas que auxiliam o desenvolvedor a fazer um trabalho com qualidade e em pouco tempo. Além disso, apresenta o projeto proposto o qual irá melhorar a qualidade de atendimento dos restaurantes, diminuindo o custo para os proprietários e possibilitando uma maior qualidade de serviço. Informando aos clientes com restrições alimentares sobre os ingredientes que possam causar reações alérgicas.

Com o sistema aqui proposto, será possível que os clientes saibam se os produtos contém ingredientes comuns de restrição, como glúten e lactose. Com a possível dispensabilidade de atendentes, o software apresenta grande chance de redução de despesas totais para o estabelecimento, e um atendimento mais preciso para os clientes.

1.1 Justificativa

Com a necessidade da informatização dos restaurantes da região e o problema com os ingredientes que causam reações alérgicas em algumas pessoas, é necessário um software, que possibilite o devido conhecimento e a visualização dos ingredientes contidos nos pratos. Isso acabará proporcionando também um atendimento mais ágil, pois com um software completo no setor de atendimento é possível sanar dúvidas sobre os produtos e facilitar o atendimento. Além de ser novidade em restaurantes será um diferencial que auxiliará a enfrentar a crise financeira.

1.2 Importância

Hoje em dia as pessoas estão cada vez mais conectadas às tecnologias e seus avanços, em alguns anos um estabelecimento onde não há tecnologia poderá gerar prejuízo para seu proprietário. Ao contrário disso, um restaurante que dispõe de um software que torna o atendimento ágil e preciso atrairá clientes.

As pessoas que possuem restrições alimentares acabam tendo problemas ao consumirem em restaurantes, por não terem a informação exata de cada um dos ingredientes contidos em cada item. O mesmo software que tornará o atendimento ágil concederá a essas pessoas o acesso à informação necessária sobre o que estarão consumindo, deixando-as mais seguras.

1.3 Objetivo Geral

Desenvolver um sistema distribuído que gerencie o setor de atendimento de um restaurante, possibilitando a visualização dos ingredientes, além de informar aos clientes se os ingredientes dos produtos contêm ou não elementos que possam causar reações alérgicas como glúten e lactose.

1.3.1 Objetivos Específicos

a) Desenvolver um *software desktop* para gerenciar todo o setor de atendimento;
b) desenvolver um *software mobile* para ser utilizado como comanda eletrônica e trazer as informações dos produtos para os clientes;
c) desenvolver uma aplicação *web* onde clientes poderão consultar os produtos e realizar reservas.

2 REVISÃO DE LITERATURA

2.1 Automação

Com a informática é possível fazer uma automação mais facilmente, atuando de forma importante para a organização e coleta de dados necessários para identificar fatores críticos, e também oferecer informações para ajudar o gerenciamento de estabelecimentos. A automação tem como objetivo melhorar os processos e assim aumentar a eficiência da empresa (REGENSTEINER, 2005).

Automação comercial consiste em tornar atividades repetitivas manuais em forma automática com o uso de sistemas e equipamentos, fazendo com que o fator falha humana não interfira no seu desempenho e melhorando o gerenciamento operacional para a empresa. Assim acaba beneficiando o cliente presente em todos os segmentos da indústria. Hoje é essencial para a eficiência dos negócios e da competitividade das empresas, trazendo mais controle sobre a gestão da empresa, e reduzindo custos e erros (GS1, AUTOMAÇÃOCOMERCIAL.ORG).

Conforme a GS1 para obter sucesso com um projeto de automação é necessário três fases, planejamento, preparação e implantação.

a) Planejamento: É a fase onde a empresa identifica as necessidades gerenciais e operacionais e assim define um ponto onde a automação pretende chegar, sendo importante o investimento, pois será idealizado um cenário com atividades e expectativas com a implantação.
b) Preparação: É onde devem ser preparadas as instalações físicas, os novos processos de trabalhos e ter definido o sistema de codificação, além de treinar e motivar o seu pessoal, bem como ter concluído os acordos com os parceiros comerciais.
c) Implantação: Nesta etapa após ter um bom desenvolvimento das etapas anteriores será a menos problemática, mas se as etapas anteriores não foram bem feitas podem causar grandes problemas para a empresa. Após a implantação, será necessário averiguar e corrigir os problemas dos processos adotados com relação às ferramentas utilizadas.

2.1.1 Objetivos e Benefícios da Automação Comercial

Nos dias de hoje o principal foco do meio empresarial é fazer da tecnologia sua aliada, para acompanhar as transformações que ocorrem cada vez mais rápido, e o mercado cada vez

mais exigente faz com que empresas mudem suas estratégias para atender as necessidades do consumidor. Também é importante estar atendo com as mudanças do mercado e com os hábitos de consumo dos clientes que mudam constantemente, com a automação funcionando perfeitamente o empresário tem tempo para focar na sua estratégia de negócio e pode dar atenção a capacitação profissional dos funcionários para um melhor atendimento. Os grandes problemas para a automação comercial ocorrer são o investimento e o impacto nos processos da empresa, além da incerteza de que os benefícios serão realmente obtidos. Porém, a automação tem benefícios que podem ser notados facilmente com as informações obtidas com a movimentação de produtos, e como consequência confere mais produtividade e confiabilidade aos processos das empresas (LEME, 2010).

No quadro 1 vemos de maneira simplificada os custos e os benefícios da automação comercial.

Quadro 1: Comparativo de custos e benefícios da automação

CUSTOS	BENEFÍCIOS
INVESTIMENTOS • Equipamentos e *softwares* • Redes de comunicação • *Layout* e mobiliário	PRODUTIVIDADE • Mais vendas devido aos melhores serviços e comodidades oferecidas • Redução de custos • Eliminação de papel
TREINAMENTO DE PESSOAL • Na implantação • De manutenção	
MANUTENÇÃO • Equipamentos • Aplicativos • Cadastros e bancos de dados	QUALIDADE • Melhoria no atendimento • Redução de erros • Aumento de segurança
IMPACTO CULTURAL • Novos processos e tarefas • Muito mais informação para ser tratada e absorvida • Resistências	COMUNICACAO VISUAL • Facilidade e atratividade para clientes • Identidade corporativa moderna e personalizada
CAPACITAÇÃO TÉCNICA • Atualização permanente • Manter equipe mínima	GESTÃO EFICAZ • Facilidade para apuração de resultados • Segurança e agilidade das informações • Eficiência na administração do fluxo de caixa

Fonte: LEME, 2010 apud GS1, 2007.

2.2 UML

2.2.1 História do UML

De acordo com Guedes (2007) UML surgiu da união de três metodologias de modelagens. Até meados de 1990, as três metodologias de modelagem orientada a objetos mais populares entre os profissionais da área de engenharia de software eram o método de OMT (*Object Modeling Technique*) de Jacobson, o método de Booch, e o método OOSE (*Object-Oriented Software Engineering*) de Rumbaugh. A Rational Software apoiou, incentivou e financiou a união dessas metodologias, com isso, logo na primeira versão grandes empresas da área de engenharia de software passaram a contribuir com sugestões para ampliar a linguagem. Em 1997 a OMG (Object Management Group) adotou UML como linguagem padrão de modelagem. Hoje na sua versão 2.3 contém muitas novidades em relação à estrutura geral da linguagem, principalmente na parte da possibilidade de se desenvolver "perfis" particulares a partir da UML e com relação à abordagem de quatro camadas.

2.2.2 Diagramas

Conforme Booch, Rumbaugh e Jacobson (2000), uma forma de entender melhor um sistema em desenvolvimento é fazer a sua modelagem, criando uma simplificação da realidade. Com o UML constroem-se modelos de blocos de construção básicos como classes, associações, interfaces, colaborações, generalizações, componentes, nós e dependências. Contudo os diagramas são formas de visualização desses blocos.

"Um diagrama é uma apresentação gráfica de um conjunto de elementos, geralmente representados como um gráfico conectado de vértices (itens) e arcos (relacionamentos)" (BOOCH, RUMBAUGH E JACOBSON, 2000, p. 89). Conforme a citação acima os diagramas são usados para visualizar no seu sistema perspectivas diferentes e assim facilitar a sua compreensão. A tabela a seguir explica um pouco sobre os principais diagramas usados.

Quadro 2: Os principais diagramas e conceitos

Nome	Conceito
Diagrama de Atividade	É um dos diagramas mais detalhistas e com maior ênfase ao nível de algoritmo da UML. Sendo possível inclusive encontrar diagramas de atividade utilizando pseudocódigo ou até mesmo uma linguagem de programação real, como Java, C ou Pascal.
Diagrama de Caso de Uso	Demonstra por meio de uma linguagem simples o comportamento externo no sistema, demonstrando as funções pelo ponto de vista do usuário.
Diagrama de Classe	Permite a visualização das classes com seus métodos e atributos e demonstra como as classes se relacionam, complementam e trocam informações entre si.
Diagrama de Componente	Mostra as dependências e os componentes do sistema, sendo que um componente representa um módulo físico do código e ele é frequentemente a mesma coisa que um pacote, mas podendo ser diferente, pois componentes representam o empacotamento físico do código.
Diagrama de Objeto	Tem como objetivo fornecer uma "visão" dos valores armazenados em um determinado momento da execução de um determinado processo do sistema, pelos objetos das classes definidas no diagrama de classes. Com o diagrama de objetos é possível criar e simular as situações pelas quais as classes passarão.
Diagrama de Sequência	Mostra a sequência de eventos que ocorrem em um determinado processo, identificando quais ações devem ser feitas entre atores e objetos envolvidos e em que ordem, ele se baseia no diagrama de caso de uso, uma vez que o caso de uso, em geral, refere-se a um processo disparado por um ator de forma generalizada.

Fonte: Fowler; Scott 2000; Guedes, 2007.

2.3 Orientação a Objeto

Orientação a objetos é um paradigma utilizado na modelagem e no desenvolvimento de software, onde contêm classes, objetos, e também outras características que podem ser visualizados no quadro 3 com os seus conceitos (SBROCCO, 2011).

Quadro 3: Conceitos de orientação a objetos

Nome	Conceito
Classe	É um conceito estático onde pode conter um conjunto de atributos e funções sendo possível encontrar apenas um desses conjuntos, também são usados como moldes para objetos.
Atributo	Representam as características da classe, contendo sempre o nome do atributo e seu tipo (inteiro, booleano, real), são considerados os elementos que diferenciam os objetos em uma mesma classe.
Método	Também conhecido como função ou operação é uma ação executada pela classe ao receber uma mensagem podendo conter ou não um parâmetro e pode retornar um valor, ele indica o comportamento da classe controlando o acesso aos atributos, em resumo executa um conjunto de instruções.
Abstração	É observar algo e conseguir capturar seu modelo conceitual e após isso representar suas características
Objeto	É uma instância de classe que contem características que são as operações (métodos) e os atributos, sendo que cada objeto único no sistema podendo organizar, manipular, criar e combinar com outros objetos.
Interface	A interface de um objeto ou classe é um conjunto de métodos que devem ser executados onde deve ser obrigatória a implementação.

Herança	Permite reaproveitar atributos e métodos de outra classe auxiliando desenvolvedores a poupar tempo de desenvolvimento e facilitando manutenções contendo uma superclasse que é a classe “pai” onde a subclasse “filha” herda seus atributos e métodos.
Encapsulamento	Ou visibilidade, específica o acesso dos seus atributos e métodos por outros objetos, é possível escolher entre esses três tipos de visibilidade. 1. Public - representado pelo símbolo “+” representa que todos os objetos podem ser acessados 2. Protected - representado pelo símbolo “#” define que só pode ser acessado pelas subclasses e pela própria classe 3. Private - representado pelo símbolo “-“ define que os elementos só podem ser acessados pela própria classe.
Polimorfismo	É associado à herança, mas ele trabalha com a redeclaração de métodos previamente herdados por uma classe fazendo com que cada objeto responda a mesma requisição de forma diferente.

Fonte: Guedes, 2007; Sbrocco, 2011;

2.4 Sistemas distribuídos

Segundo Tanenbaum e Van Steen (2007) um sistema distribuído é um sistema de computadores autônomos que trabalham em conjunto fazendo parecer que é um único sistema coerente, onde contém aplicações diferentes interligadas, e quando adequadamente projetados podem ser ampliados com facilidade, mas necessitando um software mais complexo além de levar a uma diminuição de desempenho.

2.5 Java

Java é uma linguagem de programação, sendo a mais utilizada ao redor do mundo atualmente na versão 1.8. Empresas ajudam no seu crescimento com novas adoções. Além de

ser uma linguagem de programação, ela é uma plataforma de desenvolvimento podendo ser utilizada para programar para múltiplas plataformas como *web*, *desktop*, *mobile*, televisão digital e cartões. Java possui uma grande comunidade sendo umas das mais fortes do mundo, facilitando muito o acesso para os iniciantes aprenderem a linguagem. Com o investimento de algumas empresas e de alguns usuários, existe uma grande variedade de frameworks facilitando muito o trabalho dos desenvolvedores e Java também roda outras linguagens como JPython e JRuby, além de sua máquina virtual que é o grande ponto forte que faz com que o código seja executado em qualquer sistema operacional como Windows, Linux e Mac (GONÇALVES, 2013).

2.5.1 Breve História da Linguagem JAVA

No início dos anos 90, estendendo-se o poder de computação de rede para as atividades da vida cotidiana era uma visão radical. Em 1991, havia um pequeno grupo de engenheiros da Sun chamado de "Green Team" que acreditava que aconteceria a “próxima onda” na computação, que seria a união de dispositivos de consumo digitais e computadores. Liderado por James Gosling, a equipe trabalhou em torno da ideia de criar um interpretador para pequenos dispositivos, facilitando a reescrita de software para aparelhos eletrônicos, como vídeo cassete, televisão e aparelhos de TV a cabo. Foi criada tal linguagem e batizada de *Oak*, pois James Gosling olhava um carvalho de seu escritório, o sistema foi desenvolvido com uma interface gráfica padronizada e batizado de *GreenOS* (FILGUEIRAS, 2015).

O primeiro projeto da Green Team foi o StarSeven, que era um controle remoto com interface *touchscreen* que contava com um assistente que ensinava o usuário a utilizar o controle. Houve uma competição para o desenvolvimento de uma tecnologia para TV a cabo interativo, a qual a Sun competiu e onde foi aplicado o StarSeven. Mesmo sendo um produto de alta qualidade, ele não foi o escolhido como vencedor, pois era muito avançado para a época e havia chance de corte de verbas para o projeto. A Sun em 1994 foi para o mercado da internet que estava começando crescer, e com isso o nome da linguagem foi trocada para Java, que foi uma homenagem à uma ilha da Indonésia de onde os Norte-Americanos importavam o café que era consumido pela equipe de James Gosling. Em 1996, a Sun Microsystems resolveu disponibilizar gratuitamente para a comunidade um kit de desenvolvimento de software, que ficou conhecido como JDK e isso fez crescer a aceitação do java em empresas e com desenvolvedores (PACIEVITCH; FILGUEIRAS, 2015).

Em 2009 a Oracle compra a Sun Microsystems após o fracasso da negociação com a IBM “A Oracle, um dos maiores fabricantes de programas e aplicativos de informática do mundo, anunciou hoje que acertou a compra da Sun Microsystems por US$ 7,4 bilhões, apenas duas semanas depois que a IBM retirou a oferta pela mesma companhia” (G1, 2009).

Figura 1: Fragmento de código Java

```
public class Song implements Parcelable {
    private long id;
    private String title;
    private String artist;
    private String path;


    public Song(long songID, String songTitle, String songArtist, String songPath) {
        id = songID;
        title = songTitle;
        artist = songArtist;
        path = songPath;
    }
```

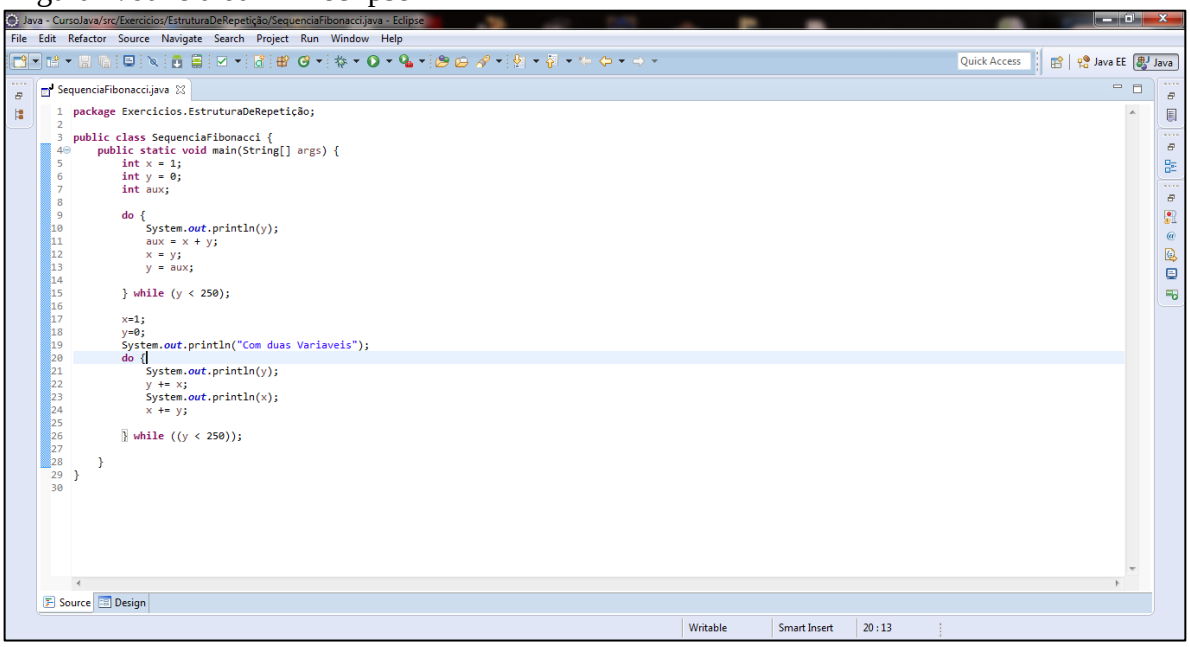
Fonte: Próprio autor

Atualmente, Java não só permeia a internet como também está presente em muitos dos aplicativos e dispositivos que acompanham nossas vidas no dia-a-dia. A partir de telefones móveis para dispositivos portáteis, jogos e sistemas de navegação para e-Business Solutions, hoje existem mais de 3 bilhões de dispositivos que rodam Java.

2.5.2 IDE Eclipse

Neste projeto foi utilizado a IDE Eclipse atualmente estando na sua versão Photon é utilizada para o desenvolvimento Java, é uma das IDEs mais utilizadas para a programação Java, ela também suporta outras várias outras linguagens como C/C++, Python, Scala e a plataforma Android ele foi desenvolvido em Java e segue o modelo *Open Source* de desenvolvimento de software, além dela tem o NetBeans da Oracle e a IntelliJ IDEA da NetBrains, que também tem uma IDE para o desenvolvimento Android.

Figura 2: Janela da IDE eclipse

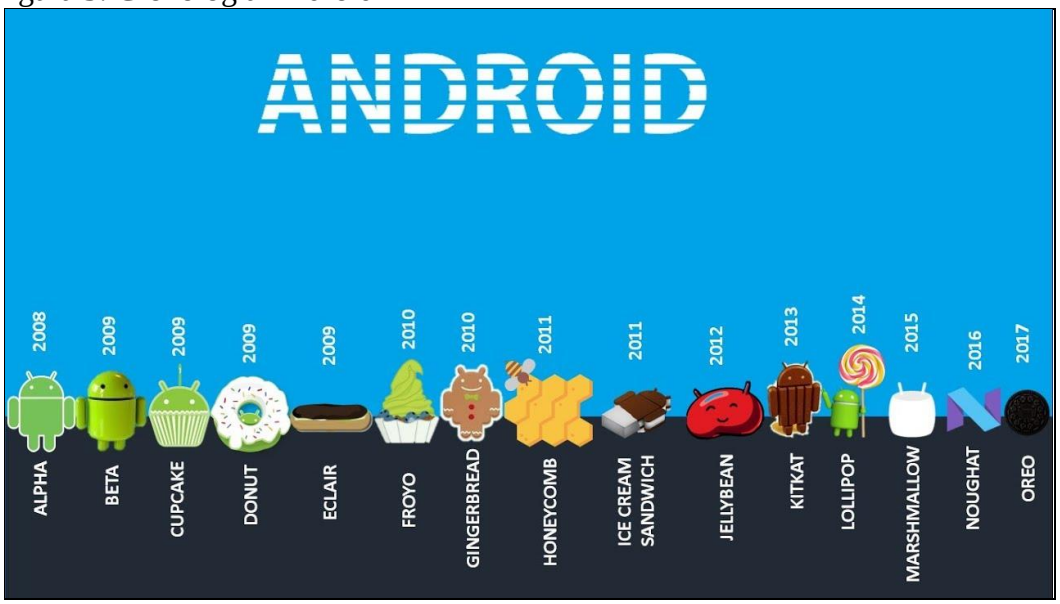


Fonte: Próprio autor.

2.6 Android

Conforme Cidral (2011) fala, o Android é o sistema operacional baseado no Linux e pertencente ao Google, que foi feito para dispositivos móveis. Além disso, ele tem uma loja virtual chamada de Google Play onde se tem aplicativos e jogos gratuitos e pagos para *smartphones* e *tablets* que utilizem o sistema operacional Android. O Android é o responsável por gerenciar todos os recursos e as tarefas de um *smartphone* ou *tablet*. O sistema operacional da Google nasceu durante a ascensão dos dispositivos *touchscreen* e de uma nova forma de desenvolvimento e distribuição de software. Não demorou muito para conseguir seu espaço no mercado, pois seus concorrentes não eram lá muito fortes. Porque eles não eram agradáveis visualmente, continham problemas de compatibilidade em dispositivos diferentes e não eram tão funcionais como o Android. Com exceção do IOS da Apple, o Symbiam, o Blackberry OS e o Windows Mobile fracassaram. A figura 3 mostra de forma simplificada a cronologia do Android, que atualmente estando na sua versão 8.0 e 9.0 sendo abertas apenas para desenvolvedores, sofreu um grande avanço no decorrer do tempo.

Figura 3: Cronologia Android

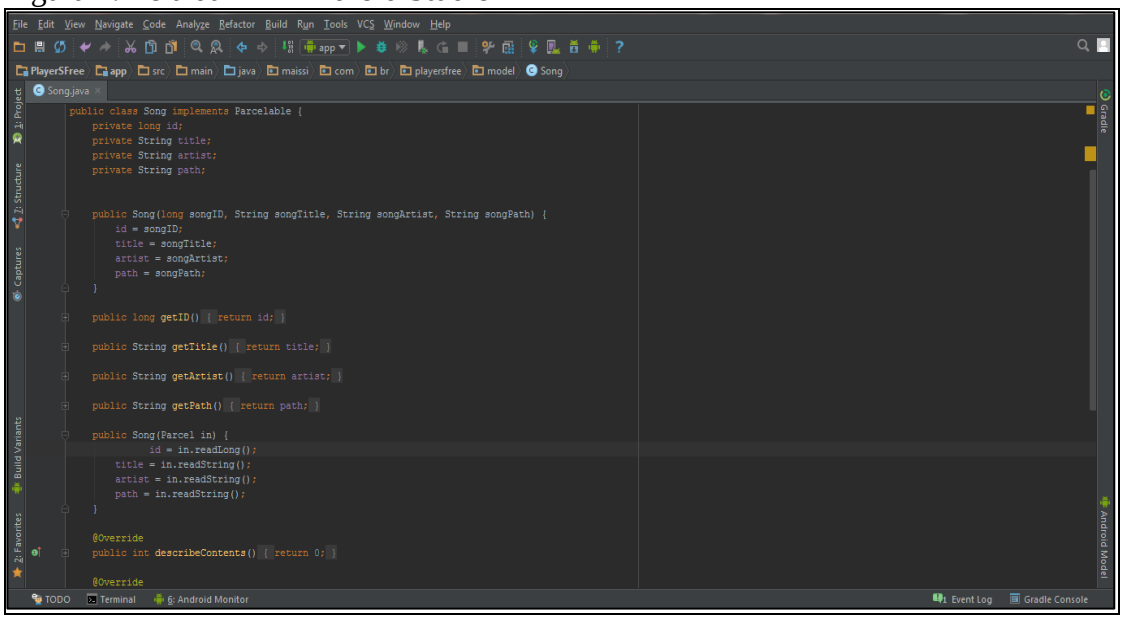


Fonte: https://i.ytimg.com/vi/olglEgEEZKo/maxresdefault.jpg

2.6.1 IDE Android

Antigamente para programar aplicativos para Android era utilizado um *plugin* na IDE eclipse que também é utilizada para programar na linguagem java, mas hoje existe o Android Studio onde atualmente está na versão 3.2.1 desenvolvida pela IDE IntelliJ com *plugin* do Android SDK (ANDROIDDEVELOPERS, 2016).

Figura 4: Tela da IDE Android Studio



Fonte: Próprio autor

2.6.2 Principais componentes

Esta tabela contém os principais componentes utilizados no Android para facilitar seu entendimento.

Quadro 4: Principais componentes do Android.

Componente	Conceito
TextView	É utilizado para mostrar texto na tela.
EditText	É utilizado para inserir texto e assim podendo ser captado para tratamento e usado em outras funções.
Button	É utilizado para executar uma ação ao utiliza-lo.
ImageButton	Assim como o *Button* executa uma ação, mas com o diferencial de conter uma imagem.
ListView	É uma lista de elementos podendo conter texto, imagem e pode executar funções ao selecionar um dos elementos.
ImageView	Utilizado para colocar uma imagem na aplicação
CheckButton	Serve como seleção múltipla é utilizada quando é possível escolher várias opções.
RadioButton	Parecido com o *CheckButton*, mas com o diferencial de que só é possível utilizar uma das opções como por exemplo escolha de sexo em cadastros.
RadioGroup	É um grupo de *radio buttons* para facilitar sua separação quando tiver mais de uma opção de escolha única

Fonte: Próprio autor.

2.7 Programação web

2.7.1 World Wide Web

Mais conhecido como WWW, é um sistema de documentos dispostos na internet que é acessado por navegadores através de links. Estes documentos são apresentados em forma de hipertexto, sendo que hipertexto são textos em formato digital podendo ser imagens, vídeos, sons, entre outros (MARTINS, 2008).

2.7.2 HTML e CSS

De acordo com Alvarez (2004), o HTML é linguagem padrão para a definição de páginas web, ela é composta de tags que são utilizadas para dar forma aos elementos apresentados na página. É uma linguagem de programação fácil de aprender permitindo qualquer pessoa criar uma página, mas usando apenas o HTML a página se torna insatisfatória em questão de design, então nisso entra o CSS.

O CSS também conhecido como folha de estilo é uma tecnologia que permite criar páginas web com um design padronizado além de poder adicionar elementos na página como tipos de letra, fundo e efeitos. CSS é escrito dentro do código HTML, mas pode ser escrito em um outro arquivo e chamado no código do HTML (ALVAREZ, 2004b).

Atualmente HTML está na sua versão 5 com muitas melhorias de sua versão anterior e o CSS está na versão 3 ambas caminham juntas no desenvolvimento web sendo necessárias suas utilizações.

Figura 5: Exemplo de código de HTML

```
1  <html>
2      <read>
3          <title> titulo da página </title>
4      </read>
5      <body>
6          aqui é onde fica o corpo da página podendo conter texto, imagens, sons
7      </body>
8  </html>
```

Fonte: Próprio autor.

Figura 6: Exemplo de código CSS

```
<STYLE TYPE="text/css">
  @font-face {
    font-family: NasalizationTeste;
    font-style:  normal;
    font-weight: normal;
    src: url(NASALIZO.eot);
  }
  @font-face {
    font-family: NeuropolTeste;
    font-style:  normal;
    font-weight: normal;
    src: url(NEUROPOO.eot);
  }
  @font-face {
    font-family: AcknowledgeTeste ;
    font-style:  normal;
    font-weight: normal;
    src: url(ACKNOWLO.eot);
  }
-->
</STYLE>
```

Fonte: http://www.maujor.com/tutorial/imagens_impfontes/importacaoCss_tela4.gif

2.7.3 JavaScript

JavaScript é uma linguagem de programação usada para criar efeitos especiais nas páginas web, tornando-a mais interativa com o usuário. Trata-se de uma linguagem que atua no lado do cliente, pois é o navegador que faz a carga de processamento e interpreta as instruções do JavaScript para executar e fazer os efeitos e as interações. É compatível com a maioria dos navegadores modernos que é o maior recurso que ela precisa, e é uma das linguagens mais atuantes no lado do cliente (ALVAREZ, 2004c).

JavaScript é uma linguagem simples e feita para que sua execução seja rápida, pessoas sem conhecimento prévio de programação terão facilidade para utilizá-la e ao praticar pode usar sua potência. Com ela é possível desenvolver páginas dinâmicas que tenham movimento, alterar sua cor, expandir um componente ao selecionar criando assim páginas interativas. Da mesma forma, podendo fazer programas como calculadoras, agendas entre outras coisas, ela também da a possibilidade de programar pequenos scripts e programas orientados a objetos. Enfim, JavaScript deixa a disposição do desenvolvedor todos os elementos que formam a página web para que possa modifica-los dinamicamente (ALVAREZ, 2004c).

2.7.4 JSP

JSP (Java Server Pages) é a tecnologia Java utilizada para criar páginas *web* utilizando Java, com ela é possível criar aplicações *web* que são executadas por vários servidores *web* e em múltiplas plataformas, sendo esse o trunfo do Java. As páginas JSP contêm códigos HTML e XML misturado com comando de *scripts* de servidor na sintaxe Java. O JSP é baseado em *servlets*, que são destinados a executar no servidor ao gerar um arquivo .jsp, nele consta a estrutura programada para ser executadas no servidor, e assim é feita uma fase de tradução antes que se tornarem funcionais fazendo com ela vire um *servlet* tornando ela em *byte codes* (ALVAREZ, 2004d).

Figura 7: Exemplo de código JSP

```
<%@page import="login.User"%><html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<%User us = new User();
String login = request.getParameter("login");
String senha = request.getParameter("senha");
boolean status = us.verificarUsuario(login,senha);

if(us.result== true){
    out.println("Login feito com sucesso " + us.nome);

}else{
    out.println("login ou senha invalidos");

}

%>
</body>
```

Fonte: https://camilolopes.files.wordpress.com/2009/05/loginjspcode1.jpg

3 METODOLOGIA

3.1 Documentação

A pesquisa bibliográfica, ou de fontes secundárias, abrange toda bibliografia já tornada pública em relação ao tema de estudo, desde publicações avulsas como boletins, jornais, revistas, livros, pesquisas, monografias, teses, materiais cartográficos e meios de comunicação orais: rádio, gravações em fita magnética e audiovisuais: filmes e televisão (MARCONI; LAKATOS, 2003).

3.2 Natureza da Pesquisa

O presente trabalho apresenta suas informações feitas pela pesquisa de campo.

Segundo Marconi e Lakatos (2003) a pesquisa de campo é aquela que procura uma resposta para um problema, procurando conseguir informações e/ou conhecimentos acerca do problema, ou para comprovar uma hipótese, e ainda, descobrir novos fenômenos ou as relações entre eles.

3.3 Tipo da Pesquisa

O tipo da pesquisa utilizada foi a exploratória onde foi verificado um problema e foi desenvolvida uma hipótese para sua solução que é o pré-projeto que consta neste trabalho.

Geralmente empregasse procedimentos sistemáticos para obtenção de observações empíricas ou para análises de dados, podendo ser feitas simultaneamente. Frequentemente obtém descrições quantitativas e qualitativas do objeto de estudo, o pesquisador deve conceituar as inter-relações entre as propriedades do fato, ambiente ou fenômeno observado (MARCONI; LAKATOS, 2003).

3.4 Técnicas de Pesquisa

Com o auxílio da observação o pesquisador identifica e obtém provas a respeito de objetivos sobre os quais os indivíduos não têm consciência, mas que orientam seu comportamento. Desempenha papel importante nos processos observacionais, no contexto da

descoberta, e obriga o investigador a um contato mais direto com a realidade. É o ponto de partida da investigação social (MARCONI; LAKATOS, 2003).

3.5 Coleta de Dados

Esta etapa da pesquisa é que se inicia a aplicação dos instrumentos elaborados e das técnicas selecionadas, a fim de se efetuar a coleta dos dados previstos. Ela é uma tarefa cansativa e toma, quase sempre, mais tempo do que se espera exigindo paciência do pesquisador, além do cuidadoso registro dos dados e de um bom preparo anterior (MARCONI; LAKATOS, 2003).

4 PROJETO

4.1 Visão Geral

Este projeto consiste em um sistema completo de automatização para restaurantes, tratando do setor de atendimento. Com ele será possível que clientes façam seus pedidos sem a necessidade de um garçom, será possível fazer o pedido via internet e também terá um sistema onde o gerente e também os funcionários poderão gerenciar todo o setor.

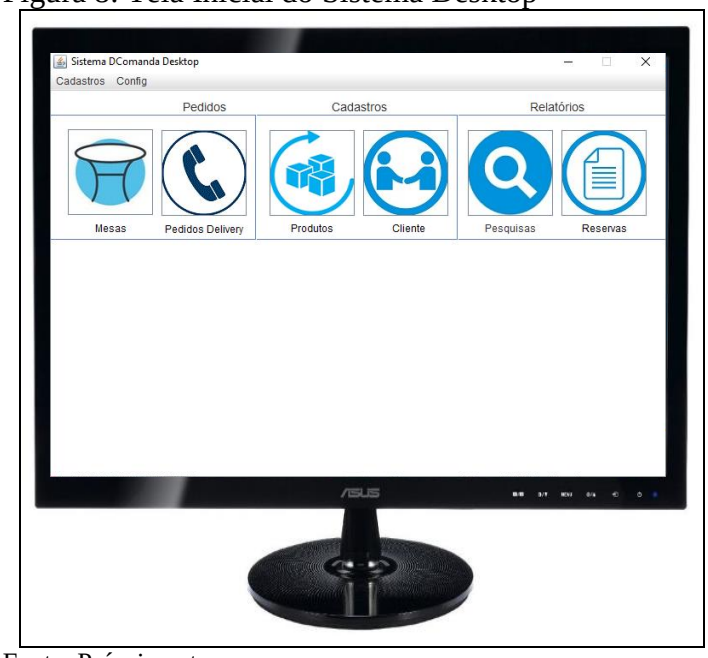
O sistema é dividido em três subsistemas sendo um deles um sistema desktop, outro é um sistema mobile, e o terceiro um sistema web.

4.2 Sistema Desktop

Ele será responsável por organizar todos os outros sistemas fazendo com que suas informações sejam armazenadas e guardadas no banco de dados, ele irá gerenciar todo o setor de atendimento do restaurante com controle de mesas, cadastro de produtos e gerenciamento dos pedidos feitos pelo sistema mobile e pelo sistema web.

Na figura 8 temos a tela inicial que é apresentada depois se fazer *login* no sistema, nela podemos acessar os pedidos em andamento, os cadastros e as reservas.

Figura 8: Tela Inicial do Sistema Desktop



Fonte: Próprio autor.

4.2.1 Módulo do Pedido e Mesas

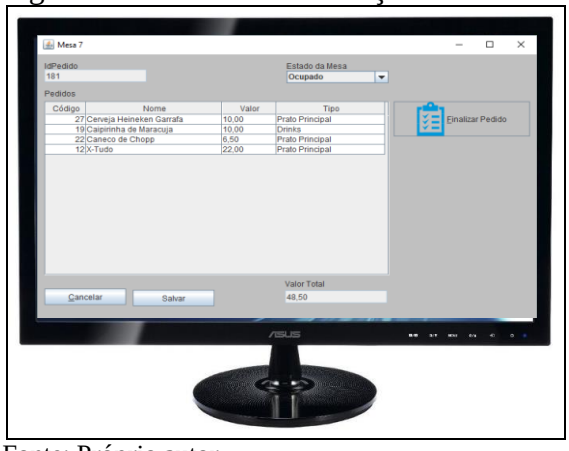
Ao acessar o botão “Mesas” é possível visualizar todas as mesas e os estados de cada uma podendo estar “livre” representada na cor verde, “ocupada” na cor vermelha e “reservada” na cor amarela, a quantidade de mesas é configurada no menu superior da tela principal escolhendo o número de mesas. O sistema gera botões automaticamente, e ao acessar a tela das mesas ele verifica o estado de cada uma, alterando a sua cor conforme o estado salvo no banco de dados. Caso tenha um pedido em aberto o sistema mobile vai salvar a informação que a mesa está sendo ocupada. Na Figura 9 mostra a tela das mesas e seus estados, o código da tela está no Apêndice A.

Figura 9: Tela com informações das mesas.

Fonte: Próprio autor.

Ao selecionar uma mesa abre outra janela onde se mostra o estado da mesa, e caso aja pedido referente a ela, mostrará as informações dos produtos pedidos e seus respectivos valores, contendo logo abaixo o valor total dos pedidos conforme mostrado na figura 10. O código dessa tela está no Apêndice B e no Apêndice F.

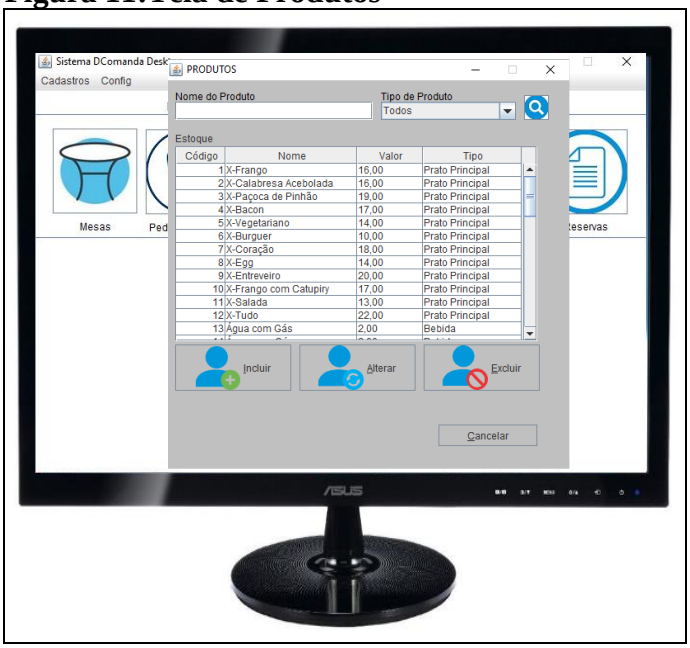
Figura 10: Tela com informações da mesa.



Fonte: Próprio autor.

4.2.2 Módulo de Produtos Ingredientes

Figura 11:Tela de Produtos



Fonte: Próprio autor.

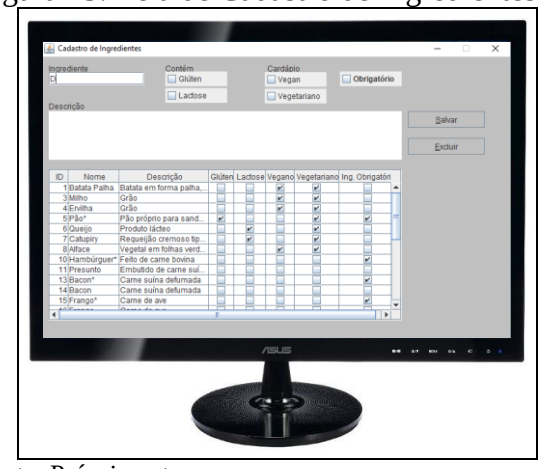
Na figura 11 temos a tela de produtos, onde são listados todos os produtos cadastrados com suas devidas informações, podendo incluir, alterar, excluir, pesquisar por nome e tipo de produto;

Figura 12: Tela de Cadastro de Produtos

Fonte: Próprio autor.

Na tela de cadastro dos produtos figura 12, é possível selecionar os ingredientes já cadastrados, além disso, tem um botão para cadastrar mais ingredientes, caso necessário, na figura 13 mostra ingredientes já cadastrados, e mostra a informação se eles contem glúten ou lactose, além te der indicadores para formar cardápios veganos e vegetarianos, o campo "obrigatório" serve para mostrar que o ingrediente são pode ser retirado do produto, pois ele é essencial para o mesmo.

Figura 13: Tela de Cadastro de Ingredientes



Fonte: Próprio autor.

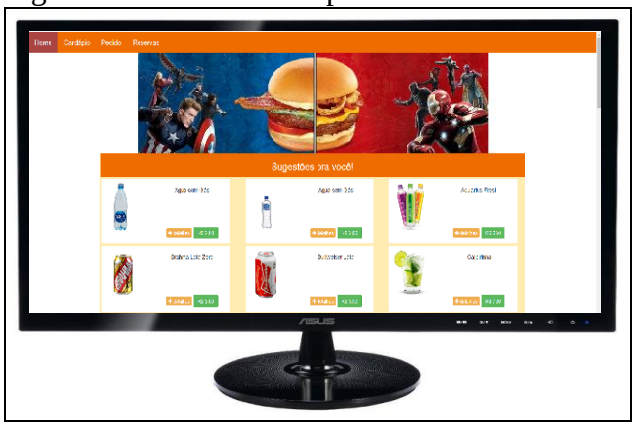
4.3 Sistema Web

O Sistema Web vai ser utilizado para usuários fazerem pedidos, reservarem mesas e verem os produtos tanto no conforto da sua casa quanto de qualquer outro lugar facilitando o acesso ao restaurante. Os clientes também terão acesso à sugestões e informação sobre os produtos, na figura 11 temos a tela inicial do sistema, e no Apêndice C têm todo o código do index do sistema, pois as outras telas são chamadas dentro dele mantendo assim o menu de navegação.

4.3.1 Cardápio e Pedido

.

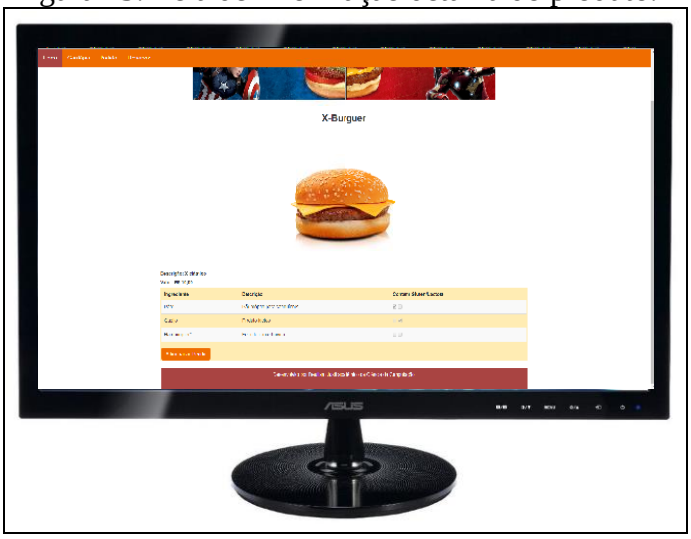
Figura 14: Tela do Cardápio



Fonte: Próprio autor.

Na figura 14 temos a tela do cardápio onde contém toda a lista de produtos separada por determinado tipo como, por exemplo, bebidas, acompanhamentos, pratos principais ou qualquer outro tipo que tenha sido cadastrado no sistema desktop e escolhido na hora de cadastrar o produto, o código da tela está no apêndice D.

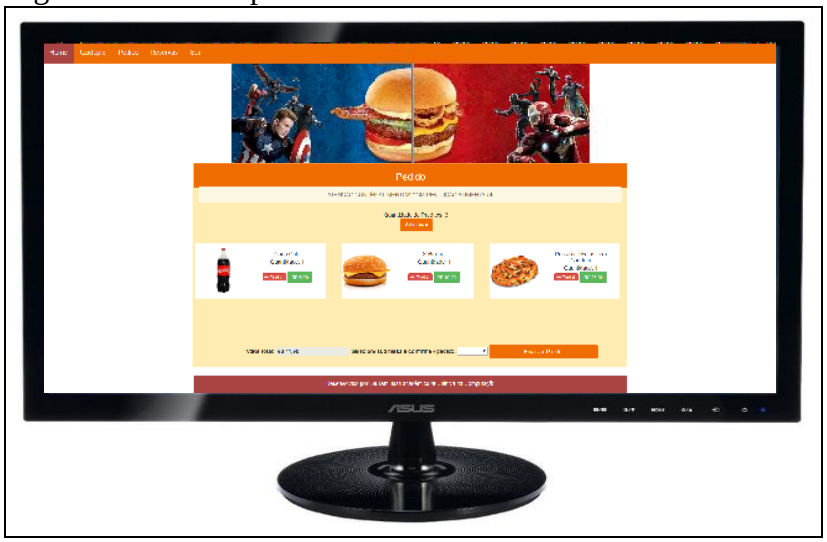
Figura 15: Tela de informação detalha do produto.



Fonte: Próprio autor.

Após o usuário clicar para ver os detalhes de um produto ele será redirecionado para a página da figura 15, onde vai haver uma imagem do prato, um botão para adicionar ao pedido, uma descrição do produto, de seus ingredientes e do valor.

Figura 16: Tela de pedido.



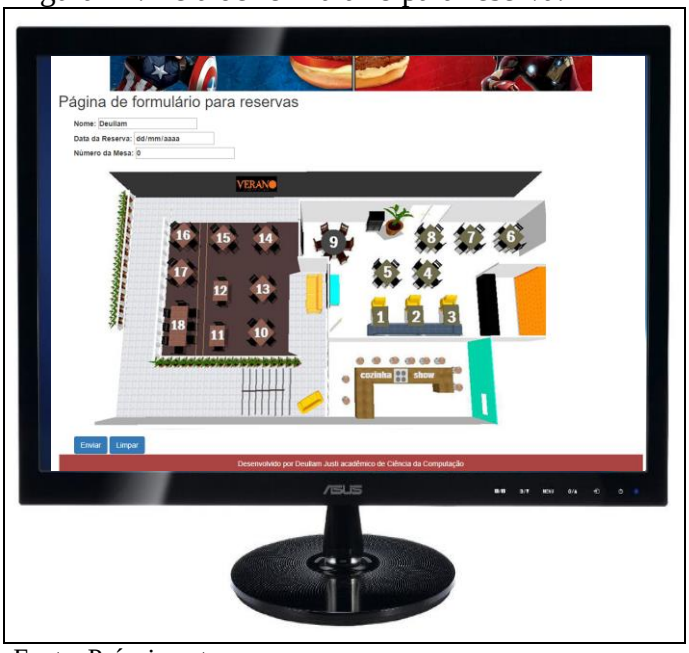
Fonte: Próprio autor.

Na figura 16 temos o pedido e os produtos previamente escolhidos pelo usuário, podendo revê-los e excluir do seu pedido, a tela contém um campo com o valor total dos produtos e dois botões um para adicionar um novo produto e outro para finalizar o pedido.

4.3.2 Reservas

Reservas podem ser feitas pelo sistema web de maneira que os usuários podem consultar suas reservas e os administradores do sistema podem consultar todas as reservas, na figura 17 mostra o formulário de reserva onde vem auto preenchido o nome do usuário todas as reservas feitas e qual o nome da pessoa que a fez, também contendo número de telefone caso o restaurante queira entrar em contato com a pessoa, no apêndice E contém todas as classes relacionadas à reserva.

Figura 17: Tela de formulário para reserva.

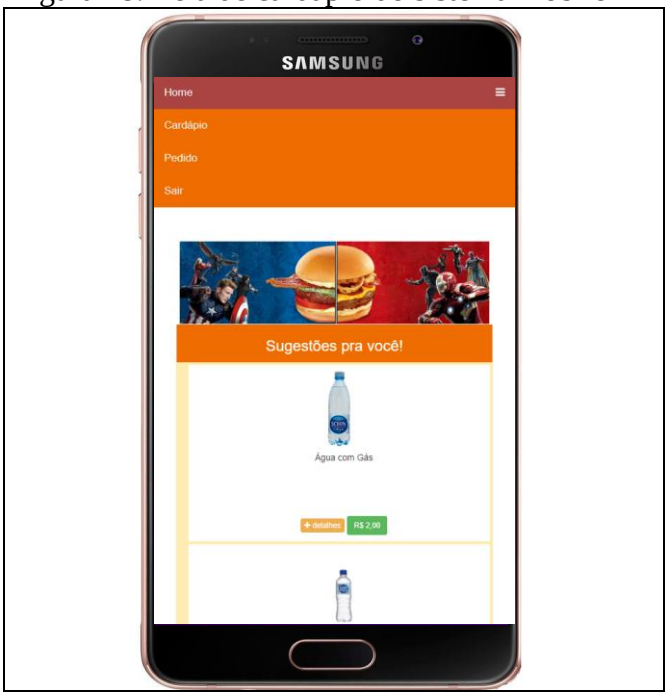


Fonte: Próprio autor.

4.4 Sistema Mobile

O sistema *mobile* será feito para rodar em *tablets* que estarão dentro do restaurante, ele basicamente será uma comanda eletrônica onde os clientes poderão fazer seus pedidos sem a necessidade de um garçom com uma interface agradável e intuitiva onde qualquer pessoa possa fazer seu pedido sem dificuldades. Apesar de ser um sistema semelhante ao do sistema web, ele não necessita de internet para rodar, podendo rodar em uma rede local. Na figura 18 temos a tela do cardápio.

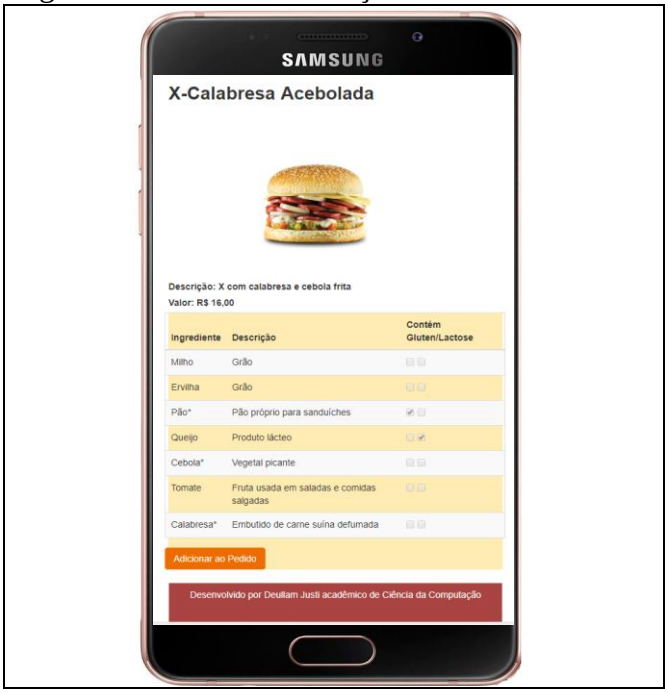
Figura 18: Tela de cardápio do sistema mobile



Fonte: Próprio autor

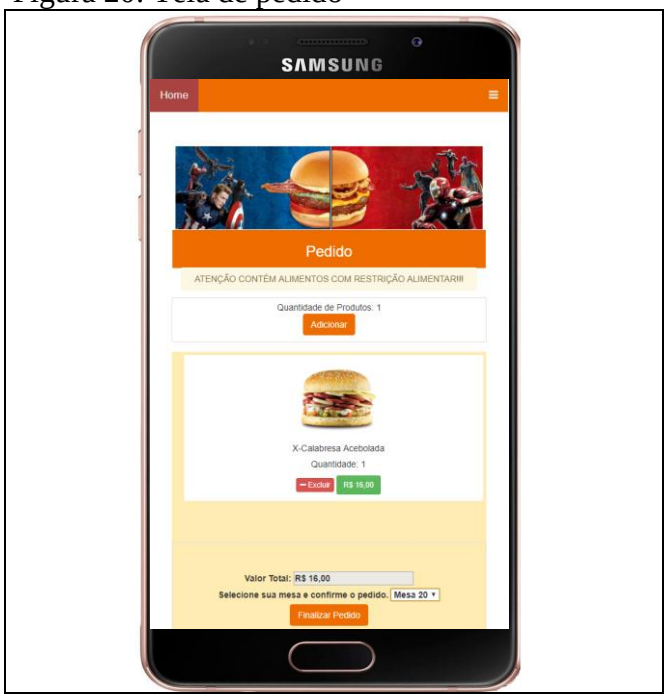
Na figura 18 temos o cardápio e na figura 19 temos a informação detalhada do produto selecionado, contendo informações sobre ele e sobre os ingredientes cadastrados especificando quais ingredientes são usados para fazer o prato.

Figura 19: Tela de informações detalhadas do Produto no app mobile



Fonte: Próprio autor

Figura 20: Tela de pedido



Fonte: Próprio autor

Na figura 20 temos o estado do pedido com seus respectivos produtos, e seus valores, logo abaixo temos o valor total e o botão para finalizar o pedido.

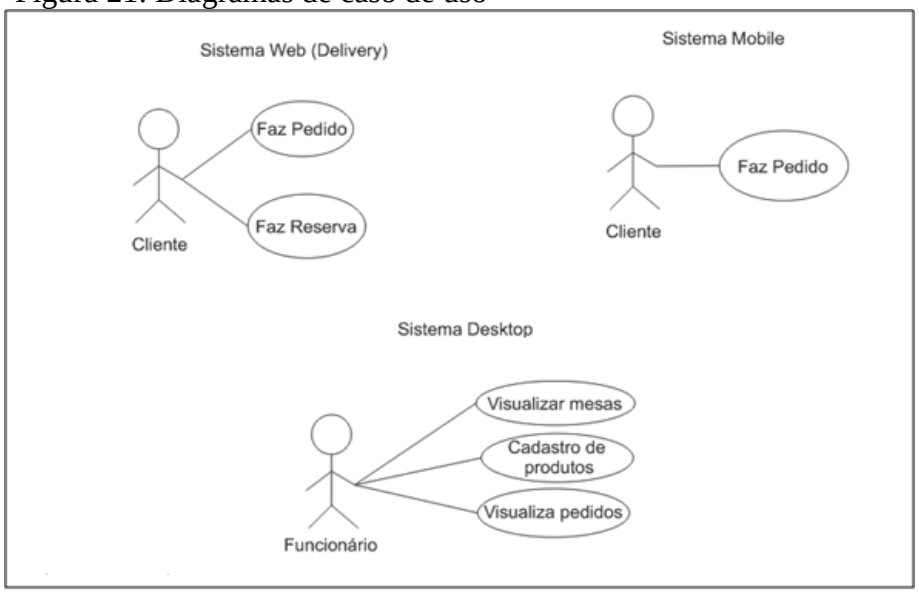
4.5 Diagramas

Os Diagramas são ilustrações sob diferentes pontos de vistas (GUEDES, 2007), a seguir veremos os diagramas de caso de uso, de atividade e de sequência para ilustrar os sistemas e suas funções.

4.5.1 Diagrama de Caso de Uso

Conforme a figura 21 têm os diagramas de caso de uso dos sistemas onde no sistema mobile o cliente apenas irá fazer o pedido, no sistema web assim como o mobile o cliente faz seu pedido, mas também pode fazer reserva no restaurante, no sistema desktop o funcionário pode fazer as funções de visualizar as mesas para gerencia-las, assim como os pedidos que estarão em andamento pelos sistemas *mobile* e *web* além de cadastrar os produtos e suas demais funções que será de extrema importância, pois será assim que os demais sistemas irão pegar as informações.

Figura 21: Diagramas de caso de uso

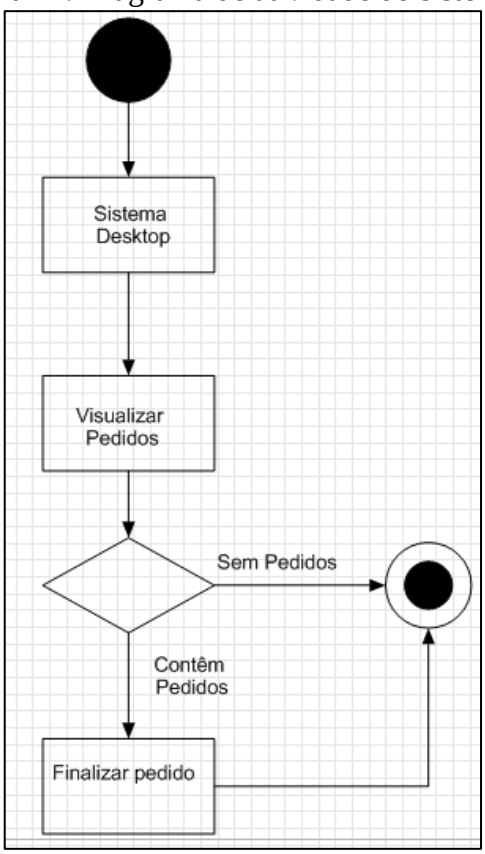


Fonte: Próprio autor

4.5.2 Diagramas de atividade

Na figura 22 contém o Diagrama de atividade do sistema desktop fazendo a finalização do pedido, ou seja, quando o cliente quer a conta para pagá-la.

Figura 22: Diagrama de atividade do sistema desktop

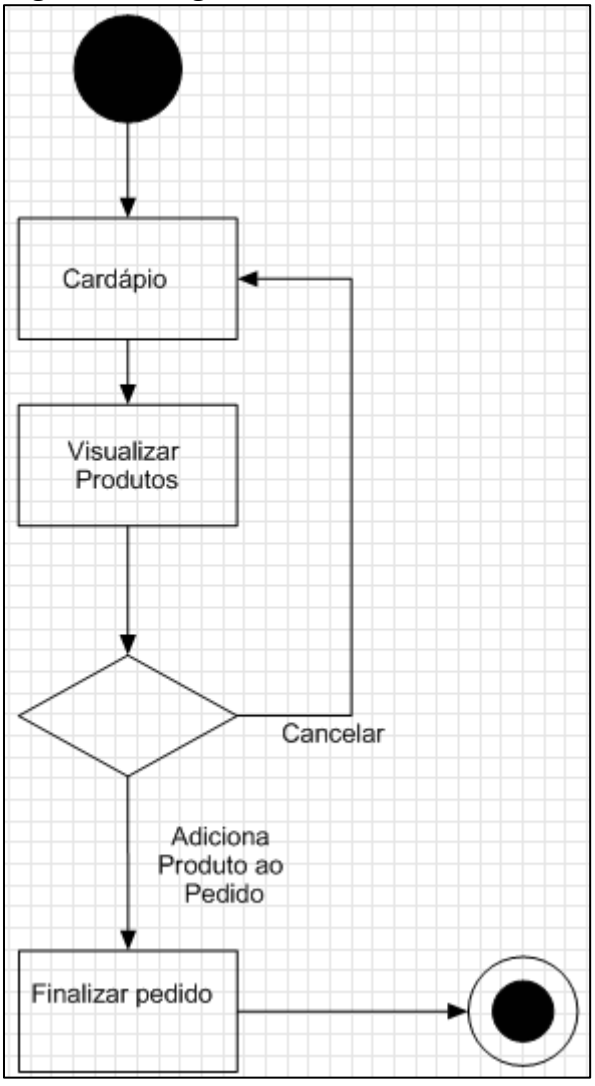


Fonte: Próprio autor

Neste diagrama o funcionário após acessar o sistema ele vai visualizar os pedidos caso haja pedido ele pode finaliza-lo e assim se encerrando a atividade, caso não haja pedidos a atividade de encerra.

Na figura 23 temos o diagrama do sistema *mobile* na atividade de fazer um pedido, nele podemos ver que ao visualizar o produto pode adicionar ele ao pedido ou cancelar e voltar ao cardápio.

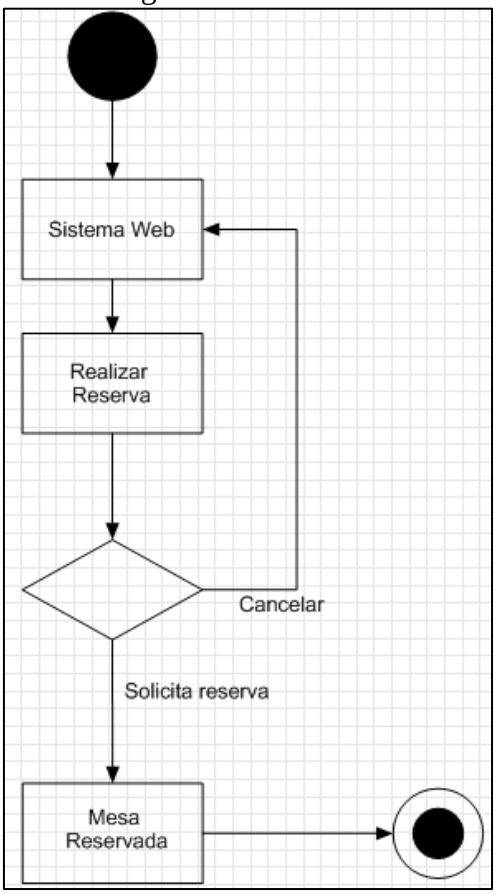
Figura 23: Diagrama de atividade do sistema mobile



Fonte: Próprio autor

Na figura 24 podemos ver que o diagrama de atividade do sistema *web* ao fazer uma reserva, sabendo que o cliente já está no *site*.

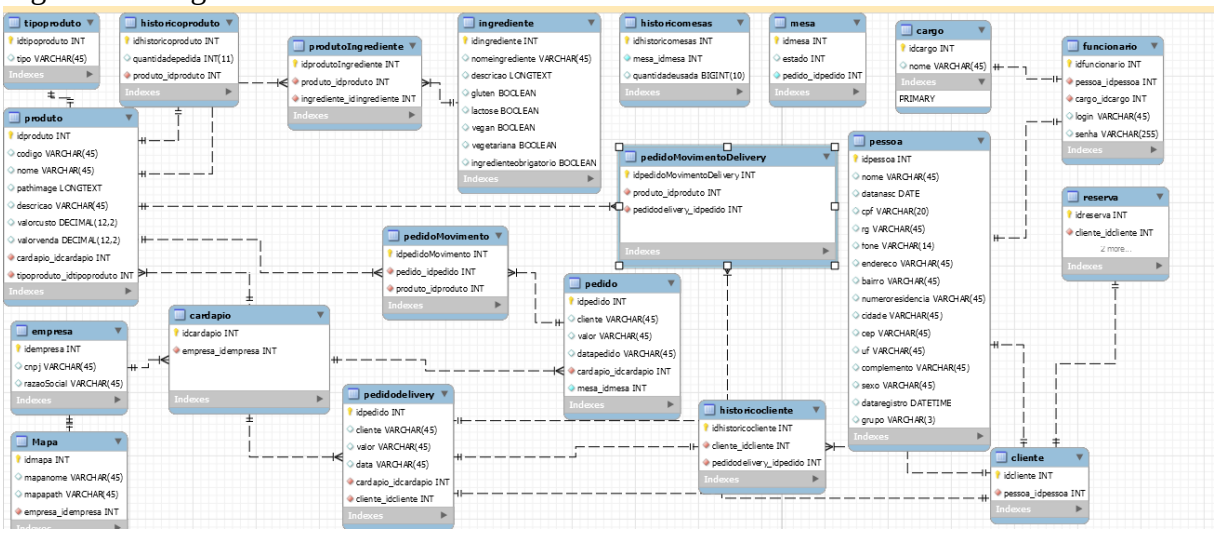
Figura 24: Diagrama de atividade do sistema Web.



Fonte: Próprio autor

Na figura 25 apresenta-se o diagrama do banco de dados onde temos todas as tabelas dos sistemas e seus atributos.

Figura 25: Diagrama do banco de dados



Fonte: Próprio autor

5 CRONOGRAMA

Para a realização do projeto proposto neste trabalho, seguiu-se o seguinte cronograma.

Quadro 5: Cronograma TCC 2

Atividades Realizadas	Março	Abril	Maio	Jun	Jul	Ago	Set	Out	Nov	Dez
Idealização do projeto e levantamento de requisitos										
Pesquisa										
Estudo de técnicas										
Especificação do protótipo										
Pré-apresentação TCC										
Entrega Final TCC										
Entrega TCC a banca avaliadora										
Desenvolvimento do Software										
Testes										
Entrega TCC II										
Defesa de Banca de TCC II										

Fonte: Próprio autor

6 TRABALHOS CORRELATOS

6.1 Restaurante Wireless

Tal trabalho visa implementar um sistema de comunicação entre clientes e o atendimento de um restaurante. Para evitar o atraso de ter que um garçom venha até a mesa, ou até de que não se encontre um garçom no momento desejado. O cliente sem sair do lugar envia um pedido diretamente para a cozinha e aguarda o pedido lhe seja entregue prontamente.

Assim como o sistema proposto nesse trabalho, ele faz a automação no setor de atendimento com a diferença que o sistema DComanda é utilizado um *tablet* e o sistema do Restaurante Wireless utilizava um display gráfico GLCD, um Transceiver e um teclado.

7 RESULTADOS

Ao final do processo de desenvolvimento foi possível: desenvolver uma aplicação desktop; desenvolver uma aplicação mobile; desenvolver uma aplicação web; desenvolver a comunicação das aplicações entre si.

Com as aplicações implantadas em um restaurante, o sistema poderá proporcionar algumas melhoras no restaurante como: economizar custos com funcionários; melhorar a qualidade de atendimento; melhorar a agilidade da entrega de pedidos; aumentar a segurança da entrega de pedidos; informar aos clientes se os produtos contêm ou não ingredientes que podem causar reações alérgicas.

8 REFERÊNCIAS

ALVAREZ, Miguel Angel. **O que é HTML** 2004a. Disponível em: <http://www.criarweb.com/artigos/7.php>. Acesso em: 14 maio 2016.

_____ **O que é CSS** 2004b. Disponível em: <http://www.criarweb.com/artigos/7.php>. Acesso em: 14 maio 2016.

_____ **O que é JavaScript** 2004c. Disponível em: <http://www.criarweb.com/artigos/7.php>. Acesso em: 15 maio 2016.

_____ **O que é JSP** 2004d. Disponível em: <http://www.criarweb.com/artigos/7.php>. Acesso em: 15 maio 2016.

ALVES, José Luiz Loureiro. **Intrumentação, Controle e Automação de Processos.** 2. ed. Rio de Janeiro: Ltc, 2013.

ANDROID DEVELOPERS**. Android Studio Release Notes**. Disponível em: <https://developer.android.com/studio/releases/index.html>. Acesso em: 12 jun. 2016.

AUTOMAÇÃOCOMERCIAL.ORG (Org.). **O QUE É AUTOMAÇÃO COMERCIAL?** Disponível em: <http://www.automacaocomercial.org/>. Acesso em: 14 jun. 2016.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML:** Guia do usuário. 10. Rio de Janeiro: Campus, 2000. 472 p. Tradução Fábio Freitas da Silva.

CIDRAL, Beline. **Afinal, o que é Android?** 2011. Disponível em: <http://www.techtudo.com.br/artigos/noticia/2011/01/afinal-o-que-e-android.html>. Acesso em: 12 jun. 2016.

FILGUEIRAS, Fellipe. **Java - A Origem.** 2015. Disponível em: <http://tableless.com.br/java-origem/>. Acesso em: 02 jun. 2016.

FOWLER, Martin; SCOTT, Kendall. **UML Essencial:** Um Breve guia para a linguagem-padrão de modelagem de objetos. 2. ed. Porto Alegre: Bookman, 2000. 169 p. Tradução Vera Pezerico e Christian Thomas.

G1. **Oracle compra Sun Microsystems por US$ 7,4 bilhões.** 2009. Disponível em: <http://g1.globo.com/Noticias/Mundo/0,,MUL1091630-5602,00-ORACLE+COMPRA+SUN+MICROSYSTEMS+POR+US+BILHOES.html>. Acesso em: 02 jun. 2016.

GONÇALVES, Otávio. **Por que java?** 2013. Elaborada por. Disponível em: <http://www.devmedia.com.br/por-que-java/20384>. Acesso em: 02 jun. 2016.

GS1, Brasil. **Perguntas Mais Frequentes.** Disponível em: <https://www.gs1br.org/faq>. Acesso em: 14 jun. 2016.

GUEDES, Gilleanes T.a. **UML 2:** Guia prático. São Paulo: Novatec Editora, 2007. 174 p.

LEME, Marcelo Luis. **Desenvolvimento de um software para automatizar as comandas de pedidos em restaurantes**. 2010. Monografia (Graduação em Engenharia de Computação) – Universidade São Francisco, Itatiba.

MARCONI, Marina de Andrade; LAKATOS, Eva Maria. **Fundamentos de metodologia cientifica.** 5. ed. São Paulo: Atlas, 2003.

MARTINS, Elaine. **O que é World Wide Web?** 2008. Disponível em: <http://www.tecmundo.com.br/web/759-o-que-e-world-wide-web-.htm>. Acesso em: 14 maio 2016.

PACIEVITCH , Yuri. **História do Java.** Disponível em: <http://www.infoescola.com/informatica/historia-do-java/>. Acesso em: 02 jun. 2016.

REGENSTEINER, Roberto J. **Elementos básicos para o planejamento da automação** do varejo, 3ª ed. São Paulo: Senac São Paulo, 2005. 108 p.

SBROCCO, José Henrique Teixeira de Carvalho. **UML 2.3:** Teoria e Prática. São Paulo: Érica, 2011. 270 p.

TANENBAUM, Andrew S.; VAN STEEN, Maarten. **Sistemas distribuídos: princípios e paradigmas**. 2. ed. São Paulo: Pearson Prentice Hall, 2007. 402 p. Tradutora Arlete Simille Marques; revisor técnico Wagner Zucchi.

Apêndice A - Código Tela Mesas

```
package br.com.unifacvest.gui;

import java.awt.Color;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.util.ArrayList;
import javax.print.attribute.standard.Media;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import br.com.unifacvest.config.Propriedades;
import br.com.unifacvest.model.Mesa;
import java.awt.Font;
import javax.swing.JSeparator;
import javax.swing.SwingConstants;
import javax.swing.UIManager;
import javax.swing.JMenuBar;
import javax.swing.JMenu;
import javax.swing.JMenuItem;
import java.awt.FlowLayout;
public class JFrameSelectMesas extends JFrame {
	private JPanel pnlMain;
	private JButton btnMain;
	private JLabel lblDesta;
	private JLabel jLabelRelatorios;
	ArrayList<JButton> botoes = new ArrayList<>();
	ArrayList<Mesa> mesas = new ArrayList<>();
	private int numeroMesa;
	public JFrameSelectMesas() {
		super();
		initialize();
	}
	private void initialize() {
		this.setTitle("Sistema DComanda Desktop");
		this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setLocationRelativeTo(null);
		this.setBounds(100, 100, 850, 500);
		this.getContentPane().setBackground(new Color(148, 156, 210));
		this.setContentPane(getPnlMain());
		this.setVisible(true);
	}
	private JPanel getPnlMain() {
		if (pnlMain == null) {
			pnlMain = new JPanel();
			pnlMain.setLayout(new FlowLayout(FlowLayout.LEFT, 5, 5));
```

```
                Propriedades.setBackground(pnlMain);
                botoes = adicionarBotoes();
                for (int j = 0; j < botoes.size(); j++) {
                        pnlMain.add(botoes.get(j));
                } verificaMesas(mesas);}
        return pnlMain;}
// Método para verificar estados das mesas
private void verificaMesas(ArrayList<Mesa> mesas) {
        for (int j = 0; j < mesas.size(); j++) {
                String mesaEstado = mesas.get(j).getEstado();
                if (mesaEstado == "Ocupada") {
                        botoes.get(j).setBackground(Color.RED);
                } else if (mesaEstado == "Reservada") {
                        botoes.get(j).setBackground(Color.YELLOW);
                } else {
                        botoes.get(j).setBackground(Color.GREEN);}}}
//método para adiciona os botões na tela
public ArrayList<JButton> adicionarBotoes() {
        int quantidade = Config.numeroMesas();
        for (int i = 0; i < quantidade; i++) {
                numeroMesa = i + 1;
                botoes.add(new JButton("Mesa " + numeroMesa));
                final int ii = i;
                botoes.get(i).addActionListener(new ActionListener() {
                        public void actionPerformed(ActionEvent arg0) {
                                new JFrameSEMesas(botoes.get(ii).getText());
                        }});}return botões;}}
```

Apêndice B - Código Tela mesa pedido

```
package br.com.unifacvest.gui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import javax.swing.DefaultListModel;
import javax.swing.Icon;
import javax.swing.ImageIcon;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JList;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTable;
import javax.swing.JTextField;
import br.com.unifacvest.config.Propriedades;
import br.com.unifacvest.db.PedidoDAO;
public class JFrameSEMesas extends JFrame {
	private JPanel pnlSECadastroProduto;
	private JButton btnCancelar;
	private JButton btnIncluir;
	private JTable table;
	private JLabel jLabelIdPedido;
	private JTextField jTextFieldNome;
	private JLabel jLabelPedidos;
	private JLabel jLabelEstado;
	private JLabel jLabelValorTotal;
	private JTextField jTextFieldValorCusto;
	private JList jListEstado;
	private JScrollPane jScrollPanePedidos;
	private JTable jTablePedidos;
	public JFrameSEMesas(String nomeMesa) {
		super();
		initialize(nomeMesa);
	}
	private void initialize(String nomeMesa) {
		this.setTitle(nomeMesa);
		this.setSize(750, 460);
		this.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
		this.setLocationRelativeTo(null);
		this.setContentPane(getPnlMain());
		this.setVisible(true);}
	private JPanel getPnlMain() {
		if (pnlSECadastroProduto == null) {pnlSECadastroProduto = new JPanel();
			Propriedades.setBackground(pnlSECadastroProduto);
			pnlSECadastroProduto.setLayout(null);
```

```
                        pnlSECadastroProduto.add(getBtnCancelar());
                        pnlSECadastroProduto.add(getBtnIncluir());
                        pnlSECadastroProduto.add(getJLabelIdPedido());
                        pnlSECadastroProduto.add(getJTextFieldNome());
                        pnlSECadastroProduto.add(getJLabelPedidos());
                        pnlSECadastroProduto.add(getJLabelEstado());
                        pnlSECadastroProduto.add(getJLabelValorTotal());
                        pnlSECadastroProduto.add(getJTextFieldValorCusto());
                        pnlSECadastroProduto.add(getJListEstado());
                        pnlSECadastroProduto.add(getJScrollPanePedidos());}
                return pnlSECadastroProduto;}
private JButton getBtnCancelar() {  if (btnCancelar == null) {
btnCancelar = new JButton("Cancelar");
btnCancelar.setBounds(10, 374, 120, 26);
btnCancelar.setMnemonic(KeyEvent.VK_C);
btnCancelar.addActionListener(new ActionListener() {
@Override
public void actionPerformed(ActionEvent arg0) {dispose();}});
        Propriedades.setFont(btnCancelar);}return btnCancelar;}
private JButton getBtnIncluir() {if (btnIncluir == null) {btnIncluir = new
JButton("IncluirItem");
btnIncluir.setIcon(new ImageIcon(JFrameSEMesas.class.getResource("/images/edit.jpg")));
btnIncluir.setBounds(524, 75, 200, 56);
Icon icone = new ImageIcon(getClass().getResource("/images/incluir.jpg"));
btnIncluir.setMnemonic(KeyEvent.VK_I);
btnIncluir.addActionListener(new ActionListener() {@Override
public void actionPerformed(ActionEvent arg0) {new JFrameCardapio();}});
Propriedades.setFont(btnIncluir);}
                return btnIncluir;}
        public JLabel getJLabelIdPedido() {if (jLabelIdPedido == null) {
                        jLabelIdPedido = new JLabel("IdPedido");
                        jLabelIdPedido.setBounds(10, 11, 150, 14); }return jLabelIdPedido;}
        public JTextField getJTextFieldNome() {
                if (jTextFieldNome == null) {jTextFieldNome = new JTextField();
                        jTextFieldNome.setBounds(10, 24, 150, 20);
                        jTextFieldNome.setColumns(10);}
                return jTextFieldNome;}
        public JLabel getJLabelPedidos() {
                if (jLabelPedidos == null) {
                        jLabelPedidos = new JLabel("Pedidos");
                        jLabelPedidos.setBounds(10, 55, 150, 14);}
                return jLabelPedidos;}
        public JLabel getJLabelEstado() {
                if (jLabelEstado == null) {
                        jLabelEstado = new JLabel("Estado da Mesa");
                        jLabelEstado.setBounds(366, 11, 148, 14);
                }
                return jLabelEstado;
        }
        public JLabel getJLabelValorTotal() {
```

```
            if (jLabelValorTotal == null) {
                jLabelValorTotal = new JLabel("Valor Total");
                jLabelValorTotal.setBounds(364, 359, 140, 14);
            }
            return jLabelValorTotal;
        }
        public JTextField getJTextFieldValorCusto() {
            if (jTextFieldValorCusto == null) {
                jTextFieldValorCusto = new JTextField();
                jTextFieldValorCusto.setBounds(364, 377, 150, 20);
                jTextFieldValorCusto.setColumns(10);
            }
            return jTextFieldValorCusto;
        }
        public JList getJListEstado() {
            if (jListEstado == null) {
                DefaultListModel model = new DefaultListModel();
                jListEstado = new JList(model);
                JScrollPane jscroll = new JScrollPane(jListEstado);
                jListEstado.setBounds(366, 26, 150, 20);
                model.add(0, "Livre");
                model.add(1, "Ocupado");
                model.add(2, "Reservado");
            }
            return jListEstado;
        }
        public JScrollPane getJScrollPanePedidos() {
            if (jScrollPanePedidos == null) {
                jScrollPanePedidos = new JScrollPane();
                jScrollPanePedidos.setBounds(10, 75, 507, 278);
                jScrollPanePedidos.setViewportView(getJTablePedidos());
            }
            return jScrollPanePedidos;
        }
        public JTable getJTablePedidos() {
            if (jTablePedidos == null) {
                jTablePedidos = new JTable();
                //Pega do banco o pedido e atualiza a tabela
                new PedidoDAO().atualizarTabela(jTablePedidos);
            }
            return jTablePedidos;
        }
}
```

Apêndice C – Código da Tela Index

```
<%--
    Document   : Index
    Created on : 28/08/2016, 20:40:06
    Author     : Deullam Justi
--%>
<%@page contentType="text/html" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>  <head>   <meta http-equiv="Content-Type" content="text/html" charset=UTF-8">
    <meta charset="utf-8">   <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta name="description" content=""> <meta name="author" content=""><link
href="https://fonts.googleapis.com/css?family=Source+Sans+Pro:200,200i,300,300i,400"
rel="stylesheet">
    <link href="https://fonts.googleapis.com/css?family=Calibri:300,300i,400,400i,600"
rel="stylesheet">
    <link href="https://fonts.googleapis.com/css?family=Sansita:400,700,900"
rel="stylesheet">
    <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
    <title> DComandaWeb </title>
    <link href="<%=request.getContextPath()%>/bootstrap/css/bootstrap.min.css"
rel="stylesheet">
    <!-- Custom styles for this template -->
    <link href="<%=request.getContextPath()%>/css/normalize.css" rel="stylesheet">
    <link href="<%=request.getContextPath()%>/bootstrap/css/starter-template.css"
rel="stylesheet">
    <link rel="stylesheet" href="<%=request.getContextPath()%>/bootstrap/css/bootstrap-
responsive.css" />
    <link rel="stylesheet" type="text/css" href="<%=request.getContextPath()%>/css/font-
awesome.min.css">
        <link rel="stylesheet" href="<%=request.getContextPath()%>/CSS/MeuEstilo.css" />
        <link rel="stylesheet" type="text/css"
href="https://fonts.googleapis.com/icon?family=Material+Icons">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
    </head>  <body>  <%if (session.getAttribute("LOGADO") != null &&
session.getAttribute("LOGADO").equals("sim")) { %>
        <header id="cabecalho" class="navbar-fixed-top">  <div class="topnav"
id="myTopnav">
  <a class="active" href="index.jsp?op=home">Home</a>
  <a href="index.jsp?op=prod">Cardápio</a>
  <a href="index.jsp?op=pedido">Pedido</a>
  <a href="index.jsp?op=reserva">Reservas</a>
  <a href="javascript:void(0);" class="icon" onclick="myFunction()">
    <i class="fa fa-bars"></i>
  </a></div><script>function myFunction() {var x =
document.getElementById("myTopnav"); if (x.className === "topnav") {
x.className += " responsive";  } else {  x.className = "topnav";  }}</script>
    </header>  <div class="banner-topo text-center hidden-md-down">
        <img class="banner-topo" src="<%=request.getContextPath()%>/Imagens/Banner.png">
    </div>  <div class="container">  <%}%><section>  <%
```

```
                    if (request.getParameter("op") != null) {
                       if (request.getParameter("op").equals("usrNew")) {
                          %><%@include file="formUsuario.jsp" %><%
                       } else if (request.getParameter("op").equals("usr")) {
                          %><%@include file="formListaUsuarios.jsp" %><%
                       } else if (request.getParameter("op").equals("reserva")) {
                          %><%@include file="formListaReserva.jsp" %><%
                       } else if (request.getParameter("op").equals("reservaNew")) {
                          %><%@include file="formReserva.jsp" %><%
                       } else if (request.getParameter("op").equals("prodNew")) {
                          %><%@include file="formProduto.jsp" %><%
                       } else if (request.getParameter("op").equals("prod")) {
                          %><%@include file="formListaCardapio.jsp" %><%
                       } else if (request.getParameter("op").equals("prodView")) {
                          %><%@include file="formProdutoView.jsp" %><%
                       } else if (request.getParameter("op").equals("prodEdit")) {
                          %><%@include file="formProduto.jsp" %><%
                       } else if (request.getParameter("op").equals("prodDel")) {
                          %><%@include file="pedido.jsp" %><%
                       } else if (request.getParameter("op").equals("pedido")) {
                          %><%@include file="formListaPedido.jsp" %><%
                       } else if (request.getParameter("op").equals("pedidoNew")) {
                          %><%@include file="formPedido.jsp" %><%
                       } else if (request.getParameter("op").equals("itemPedidoNew")) {
                          %><%@include file="ProdutoView.jsp" %><%
                       } else if (request.getParameter("op").equals("home")) {
                          %><%@include file="formListaCardapio.jsp" %><%
                       } else if (request.getParameter("op").equals("login")) {
                          %><%@include file="formLogin.jsp" %><%
                       } else {   %><%@include file="formLogin.jsp" %><%
                       }
  } else{ %><%@include file="formListaCardapio.jsp" %><%}%> </section> <br>
      <footer class="footer">   <div class="col-md-7 text-center p-0 mt-1 rodape ">
      <div style="background-color: #a94442;color: #fff" class="p-3 rodape">
        <h5 class="m-0"> Desenvolvido por Deullam Justi acadêmico de Ciência da
Computação</h5>
      </div>
      </div>
           </footer>
      </div>
            <!-- Bootstrap core JavaScript
   ================================================== -->
   <!-- Placed at the end of the document so the pages load faster -->
   <script src="https://code.jquery.com/jquery-2.1.3.min.js"></script>
   <script src="<%=request.getContextPath()%>/bootstrap/js/bootstrap.min.js"></script>
   <script src="<%=request.getContextPath()%>/bootstrap/js/vendor/holder.min.js"></script>
   </body>
</html>
```

Apêndice D – Código do cardápio

```
<%--

    Document   : formListaCardapio

    Created on : 18/09/2015, 19:25:56

    Author     : Deullam Justi

--%>

<%@page import="br.com.unifacvest.model.Produto"%>

<%-- valida se usuário esta autenticado --%>

<%@include file="validalogin.jsp"%>

<%@page import="bd.OperaBD" %>

<%@page import="java.sql.*" %>

<% // cria a instancia da classe de conexão com o banco de dados

    OperaBD dados = new OperaBD();

    Produto p = new Produto();

    Integer idPedido = 0;

    PreparedStatement prepara = dados.preparaSQL("SELECT * FROM produto ORDER BY
nome");  ResultSet rs = prepara.executeQuery();%>

<form name="formListaCardapio" method="post" action="formProdutoView.jsp">

    <div class="col-md-12 text-center p-0 mt-1 sugestoes ">

        <div style="background-color: #ef6c00;color: #fff" class="p-3 sugestoes">

   <h3 class="m-0">Sugestões pra você!</h3>   </div>  </div>

    <div class="container-fluid" style="background-color: #ffecb3;">

 <div class="row"> <% while (rs.next()) { .setPathImage(rs.getString("pathimage"));

out.println("<div class='col-md-4 col-12 pt-3'>");

out.println("<div style='background-color: #fff;margin: 5px;padding: 10px; border-radius:
2px;' class='produto'>");

out.println("<div class='row'>"); out.println("<div class='col-md-5'>");

out.println("<a href='javascript:void(0);' onclick='$(" + rs.getString("idProduto") +
").click();'>");
```

```
out.println("<img class='img-fluid' src=" + p.getPathImage() + " alt='" + p.getNome() + "'
title='" + p.getNome() + "' /></a>");

out.println("</div>"); out.println("<div class='col-md-7 text-center'>");

out.println("<div class='pt-5'>"); out.println("<div class='name-equal-height' style='height:
100px;'>"); out.println("<h5 class='produto-nome'>" + rs.getString("nome") + "</h5>");

out.println("</div>"); out.println("<div class='text-center pt-3'>");

out.println("<a id='" + rs.getString("idProduto") + "' class='btn btn-sm btn-warning btn-xs
tooltipx modaalx-ajax tooltipstered' style='color: #fff;' href='index.jsp?op=prodView&id=" +
rs.getString("idProduto") + "" data-modaal-
scope='modaal_15427349917980dbb7095b68b6'>");

out.println("<i class='fa fa-plus'></i> detalhes"); .println("</a>");

out.println("<a href='#' class='btn btn-success btn-sm' style='color: #fff;'>");

out.println("R$ " + rs.getBigDecimal("valorvenda").toString().replace(".", ",") + " </a>");

out.println("</div>"); .println("</div>"); .println("</div>"); .println("</div>");

out.println("</div>"); .println("</div>"); }

%>    <input type="hidden" name="idProduto" value="" />

   <input type="hidden" name="nomeProduto" value="" />

   <input type="hidden" name="descricao" value="" />

   <input type="hidden" name="pathimage" value="" />

   <input type="hidden" name="valorvenda" value="" /> <div>    </div> </form>
```

Apêndice E – Códigos das classes usadas em Reservas

FormListaReserva.jsp

```
<%--
    Document   : formListaReserva
    Created on : 18/09/2016, 19:25:56
    Author     : Deullam Justi
--%>
<%-- valida se usuário esta autenticado --%>
<%@include file="validalogin.jsp"%>
<%@page import="bd.OperaBD" %>
<%@page import="java.sql.*" %>
<% // cria a instancia da classe de conexão com o banco de dados
    OperaBD dados = new OperaBD();
    PreparedStatement prepara = dados.preparaSQL("SELECT * FROM reserva ORDER BY
datareserva");
    ResultSet rs = prepara.executeQuery();%>
<form name="formListaReserva" method="post" action="">
    <h1>Reservas</h1>
    <%="Quantidade de registros: " /*+ dados.qtdaRegistros(rs) */%>
    <a href="index.jsp?op=reservaNew" class="componenteDireita">Adicionar</a>
    <br/><br/><table><thead>
            <tr>
                <td>Código</td>
                <td>Nome</td>
                <td>Data Reserva</td>
                <td>Telefone</td></tr></thead><tbody><%while (rs.next()) {
                    out.println("<tr>");
                    out.println("<td>" + rs.getInt("idreserva") + "</td>");
                    out.println("<td>" + rs.getString("nome") + "</td>");
                    out.println("<td>" + rs.getDate("datareserva") + "</td>");
                    out.println("<td>" + rs.getString("telefone") + "</td>");
out.println("</tr>");
                }

            %>
        </tbody>
    </table>
</form>
```

FormReserva.jsp

```
<%--
    Document   : formReserva
    Created on : 18/09/2016, 20:41:33
    Author     : Deullam Justi
--%>
<%@include file="validalogin.jsp" %>

<%    String nome = "", telefone = "";
```

```
%>

<p  style="font-size: 30px">P&aacute;gina de formul&aacute;rio para reservas

<form action="reserva.jsp" method="post">
    <input type="hidden" name="fn" value="new" />
    <label for="nome">Nome:
        <input type="text" name="nome" id="nome" value="<%= nome%>"/>
    </label>
    <br>
        <label for="datareserva">Data da Reserva:
        <input type="date" name="datareserva" id="datareserva"  />
    </label>
    <br>
    <label for="telefone">Telefone:
        <input type="text" name="telefone" id="telefone" value="<%= telefone%>"/>
    </label>
    <br>
    <input type="submit" name="botaoEnviar" id="botaoEnviar" class="botao"
value="Enviar"/>
    <input type="reset" name="botaoLimpar" id="botaoEnviar" class="botao"
value="Limpar"/>

</form>
```

Reserva.jsp

```
<%@include file="validalogin.jsp"%>
<%@page import="bd.OperaBD" %>
<%@page import="java.sql.*" %>
<%    OperaBD dados = new OperaBD();

    //opção para salvar um novo usuário na base de dados
    if (request.getParameter("fn").equals("new")) {

        String sql = "INSERT INTO reserva(nome,datareserva,telefone)VALUES(?,?,?);";
        PreparedStatement prepara = dados.preparaSQL(sql);

        prepara.setString(1, request.getParameter("nome"));
        prepara.setString(2, request.getParameter("datareserva"));
        prepara.setString(3, request.getParameter("telefone"));
        prepara.executeUpdate();
        out.println("<script>document.location.href='index.jsp?op=reserva';</script>");
    }
%>
```

Apêndice F – PedidoDAO

```
package br.com.unifacvest.db;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.logging.Level;
import java.util.logging.Logger;

import javax.swing.JTable;

import br.com.unifacvest.model.Pedido;
import net.proteanit.sql.DbUtils;

public class PedidoDAO implements DAO<Pedido> {

	private Connection con;

	public PedidoDAO() {
		// TODO Auto-generated constructor stub
		con = new FabricaConexao().getConexao();
	}

	@Override
	public void adiciona(Pedido p) {
		// TODO Auto-generated method stub
		String sql = "INSERT INTO pedidomovimento (idpedido, idproduto,
valortotal) VALUES(?,?,?)";
		try {
			PreparedStatement ps = con.prepareStatement(sql);
			ps.setInt(1, p.getIdProduto());
			ps.setBigDecimal(2, p.getValorTotal());
			ps.execute(); // para pesquisa executequery.
			ps.close();

		} catch (SQLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}

	@Override
	public void alterar(Pedido p) {
		String sql = "UPDATE pedidomovimento SET idproduto=?, valortotal=?
WHERE idpedido=?";
		try {
			PreparedStatement ps = con.prepareStatement(sql);
```

```
                ps.setInt(1, p.getIdProduto());
                ps.setBigDecimal(2, p.getValorTotal());
                ps.setInt(3, p.getIdPedido());
                ps.execute(); // para pesquisa executequery.
                ps.close();

            } catch (SQLException e) {
                e.printStackTrace();
            }

        }

        @Override
        public void excluir(int id) {
            // TODO Auto-generated method stub

            String sql = "DELETE FROM pedidomovimento WHERE idpedido=?";
            try {
                PreparedStatement ps = con.prepareStatement(sql);
                ps.setInt(1, id);
                ps.execute();
                ps.close();
            } catch (SQLException e) {
                e.printStackTrace();
            }

        }

        public void atualizarTabela(JTable j) {
            String sql = "SELECT pe.idProduto, pr.nome, pr.valor FROM pedido pe
INNER JOIN produtos pr on pe.idProduto = pr.idProduto ORDER BY pr.valor";

            PreparedStatement ps;
            try {
                ps = con.prepareStatement(sql);
                ResultSet re = ps.executeQuery();
                j.setModel(DbUtils.resultSetToTableModel(re));
                re.close();
                ps.close();
            } catch (SQLException ex) {
                Logger.getLogger(PedidoDAO.class.getName()).log(Level.SEVERE,
null, ex);
            }
        }

        @Override
        public void pesquisa(JTable j, String f) {
            // TODO Auto-generated method stub

        }}
```