CENTRO UNIVERSITÁRIO DE BRASÍLIA -UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

Marco Aurélio de Aguiar Santos

SISTEMA PARA CONTROLE DE PEDIDOS EM RESTAURANTES

Orientador: MsC. Prof. Francisco Javier de Obaldia Diaz

Brasília

Novembro, 2014

Marco Aurélio de Aguiar Santos

SISTEMA PARA CONTROLE DE PEDIDOS EM RESTAURANTES

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Brasília

Novembro, 2014

Marco Aurélio de Aguiar Santos

SISTEMA PARA CONTROLE DE PEDIDOS EM RESTAURANTES

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas - FATECS.

_____

Prof. Abiezer Amarilia Fernandes
Coordenador do Curso

Banca Examinadora:

_____

Prof. Francisco Javier De Obaldia, Mestre
Orientador

_____

Prof. Flávio A. Klein, Mestre
UniCEUB

_____

Prof. Luís Cláudio Lopes de Araújo, Mestre
UniCEUB

_____

Prof. Luciano Henrique Duque, Mestre.
UniCEUB

AGRADECIMENTOS

Agradeço a Deus por me dar saúde e forças, sem as quais não teria condições de concluir este projeto.

Aos meus pais, que sempre me apoiam nas minhas escolhas. Por me terem dado uma excelente educação e me ensinarem a nunca desistir das minhas metas. Sem vocês nunca teria chegado tão longe.

Aos professores do curso de engenharia da computação, pela paciência e disposição para ensinar e esclarecer as dúvidas. Graças a vocês adquiri os conhecimentos necessários para a construção deste projeto.

Aos meus colegas de curso, que me auxiliaram nos momentos de dificuldade e tornaram as minhas experiências na faculdade mais agradáveis.

SUMÁRIO

LISTA DE FIGURAS.....VI
LISTA DE TABELAS..... VII
LISTA DE SÍMBOLOS E ABREVIATURAS..... VIII
RESUMO..... IX
ABSTRACT..... X
CAPÍTULO 1 - INTRODUÇÃO.....11
1.1 - Apresentação do Problema 11
1.2 - Objetivos.....15
1.3 - Justificativa e Importância do Trabalho.....16
1.4 - Escopo do Trabalho.....16
1.5 – Resultados Esperados.....17
1.6 - Estrutura do Trabalho.....17
CAPÍTULO 2 - BASES METODOLÓGICAS PARA RESOLUÇÃO DO PROBLEMA 18
2.1 - Microcontrolador.....18
2.1.1 - Arduino.....18
2.2 - Linguagens.....19
2.2.1 - HTML.....19
2.2.2 - JavaScript.....20
2.2.3 - CSS.....20
2.2.4 - XML.....21
2.2.5 - PHP.....21
2.3- Biblioteca JQuery.....22
2.4 – AJAX.....22
2.5 - Protocolos.....23
2.5.1 - HTTP.....23
2.6 - Servidor.....23
2.6.1 - Apache.....23
2.7 - Cliente.....24
2.8 – Arquitetura de Software.....24
2.8.1 - MVC.....24
2.9 – Banco de Dados.....25

2.9.1 - MySQL.....25
2.10 – Componentes Físicos.....25
2.10.1 – *Buzzer*.....25
2.10.2 – LED.....26
2.11 - *Tablet*.....26
CAPÍTULO 3 – MODELO DE SOLUÇÃO PROPOSTO.....27
3.1 – Apresentação Geral do Modelo de Solução.....27
3.2 – Descrição da Solução.....28
3.2.1 – Estruturação do sistema.....28
3.2.2 – Banco de dados.....30
3.2.2.1 – Estrutura do banco de dados.....30
3.2.2.2 – Relacionamento entre tabelas.....32
3.2.3 – Implementação do *hardware*.....32
3.2.4 – Página de *login*.....34
3.2.5 – Interface que implementa o perfil cliente.....36
3.2.6 – Interface que implementa o perfil *cozinha*.....41
3.2.7 – Interface que implementa o perfil *caixa*.....43
3.2.8 – Interface que implementa o perfil *admin*.....44
CAPÍTULO 4 – APLICAÇÃO PRÁTICA DO MODELO DE SOLUÇÃO.....47
4.1 – Resultados.....47
4.2 - Apresentação da Área de Aplicação da Solução.....48
4.3 – Dificuldades Encontradas.....48
4.4 – Custo do projeto.....49
CAPÍTULO 5 – CONSIDERAÇÕES FINAIS.....51
5.1 – Conclusões.....51
5.2 – Sugestões para Trabalhos Futuros.....52
REFERÊNCIAS BIBLIOGRÁFICAS.....53
APÊNDICE.....55
Apêndice A – Código do Microcontrolador Arduino.....55
Apêndice B – Comandos de Criação do Banco de Dados.....57
Apêndice C – Código do Sistema.....59

LISTA DE FIGURAS

Figura 1.1 – Dispositivo móvel para atendimento ao cliente do restaurante Mangai.....13
Figura 1.2 – Sistema de Longo Alcance.....14
Figura 2.1 – Microcontrolador Arduino Uno.....19
Figura 3.1 – Visão geral da solução proposta.....28
Figura 3.2 - Organização da estrutura do sistema.....28
Figura 3.3 – Diagrama Entidade Relacionamento do banco de dados.....30
Figura 3.4 – Esquemático do circuito de alerta de pedidos.....33
Figura 3.5 – Ligação entre o Arduino e os componentes.....34
Figura 3.6 – Página de *login*.....34
Figura 3.7 – Formulário de recuperação de senha.....35
Figura 3.8 – Formulário de cadastro.....36
Figura 3.9 – Página de cardápio.....37
Figura 3.10 – Formulário de detalhamento do pedido.....38
Figura 3.11 – Coluna de pedidos.....38
Figura 3.12 – Formulário de avaliação do atendimento.....39
Figura 3.13 – Página de dúvidas.....40
Figura 3.14 – Página de alteração de senha.....40
Figura 3.15 – Página de lista de pedidos.....41
Figura 3.16 – Página de lista de dúvidas.....42
Figura 3.17 – Página cardápio.....43
Figura 3.18 – Página de contas para fechamento.....43
Figura 3.19 – Página de *admin*istração de usuários.....45
Figura 3.20 – Página de lista de avaliações.....45
Figura 4.1 – Tempo gasto no pedido.....47

LISTA DE TABELAS

Tabela 4.1 – Custo estimado dos dispositivos.....49

LISTA DE SIGLAS E ABREVIATURAS

AJAX - *Asynchronous Javascript and XML* - *Javascript* Assíncrono e XML

CSS - *Cascanding Style Sheets* – Folha de Estilos em Cascata

DER – Diagrama Entidade Relacionamento

HTML – *Hypertext Markup Language* – Linguagem de Marcação de Hipertexto

HTTP – *HyperText Transfer Protocol* – Protocolo de Transferência de Hipertexto

LED - *Light Emitting Diode* - Diodo Emissor de Luz

MVC - *Model-View-Controller* - Modelo-Visão-Controlador

PHP – *Hypertext Preprocessor* – Preprocessador de Hypertexto

SGBD – Sistema Gerenciador de Banco de Dados

SQL – *Structure Query Language* – Linguagem Estrutura de Consulta

TI – Tecnologia da Informação

USB – *Universal Serial Bus* – Barramento Serial Universal

XML – *eXtensible Markup Language* – Linguagem Extensível de Marcação

RESUMO

Esse projeto propõe o desenvolvimento de um sistema para gerenciar, controlar e automatizar pedidos de clientes em restaurantes. Foram criados quatro tipos de usuários para o sistema: *cliente, cozinha, caixa* e *admin*. Os clientes terão acesso ao cardápio por meio de um *tablet*, através do qual irão solicitar seus pedidos, acompanhar o valor da conta, etc. Os funcionários serão divididos nos usuários do tipo *cozinha* e *caixa*. Cada um deles ficará responsável por determinadas tarefas. O funcionário *cozinha* visualizará os pedidos dos clientes, receberá um aviso sonoro sobre a chegada do pedido através de um microcontrolador Arduino e atenderá aos pedidos dos clientes. O funcionário *caixa* atenderá as solicitações dos clientes referentes ao fechamento da conta. Ambos os tipos de funcionários acessarão o sistema por meio de um *desktop*. Além disso, haverá um usuário *admin* (que será o gerente ou o próprio dono do restaurante), o qual será responsável por gerenciar os dois tipos anteriores. O contato entre cliente e garçom será mínimo, sendo que este se encarregará, em boa parte do tempo, de trazer os pedidos que foram solicitados. O objetivo principal é oferecer um meio de tornar os pedidos mais céleres, já que o cliente, com o auxílio do sistema, enviará o pedido direto à cozinha, não necessitando que um garçom anote e comunique o pedido aos responsáveis pelo preparo. O modelo de solução proposto será explicado em detalhes e também será feita uma análise acerca dos estabelecimentos nos quais este projeto pode melhor ser aplicado. Ao final serão mostradas as conclusões e sugestões para trabalhos futuros.

Palavras-chave: automatização em restaurantes, Arduino, arquitetura MVC, banco de dados MySQL.

ABSTRACT

This project here proposes the development of a system to manage, control and automate customer orders in restaurants. Four types of users were created for the system: client, kitchen, pay-box and admin. Customers will have access to the application via a tablet, through which they will request their order, check the value of the account, etc. Employees will be divided into types kitchen and pay-box. Each one will be responsible for certain tasks. The employee kitchen will see customer orders, will receive an audible warning about the arrival of the orders through an Arduino microcontroller and meet customer orders. The employee pay-box will meet customer requests relating to the closing of the account. Both employees will access the system via a desktop. In addition there will be an admin user (representing the manager or the owner of the restaurant itself), which will be responsible for managing the two previous users. The contact between customer and waiter will be minimal, and the waiter will take care, in good part, of bringing the orders that were requested. The main objective is to offer ways of making a most rapid order, as the client, with the system's assistance, send the request directly to the kitchen, not requiring that a waiter write down and communicate the request to those responsible for preparing. The proposed solution model will be explained in detail and also an analysis about the establishments in which this project can best be applied. At the end are shown the conclusions and suggestions for future work.

Keywords: automation in restaurants, Arduino, MVC architecture, MySQL database.

CAPÍTULO 1 – INTRODUÇÃO

1.1 – Apresentação do Problema

Clientes de estabelecimentos como bares e restaurantes possuem certas expectativas quanto à qualidade do serviço e do atendimento oferecido no local. O não cumprimento destas expectativas pode prejudicar a imagem do estabelecimento, resultando na perda de muitos clientes e, na pior das hipóteses, na falência da empresa. Garantir que o cliente está sendo atendido da melhor maneira possível é, dessa forma, uma preocupação constante dos donos e gerentes. Preocupação esta que é agravada quanto maior for o número de clientes que está aguardando por atendimento, visto que este atendimento pode levar um tempo prolongado por estar sujeito à disponibilidade de garçons, causando incômodo ao cliente.

Diante do problema apresentado, os donos e gerentes se deparam com a seguinte questão: como corresponder às expectativas dos clientes de maneira ágil e eficiente, independentemente da quantidade de pessoas que estão esperando ser atendidas.

Segundo o psicólogo americano Abraham Maslow (1908-1970), as necessidades dos seres humanos obedecem a uma hierarquia, sendo elas (do nível mais baixo para o mais alto): fisiológicas, de segurança, sociais, de estima e de auto-realização. As necessidades humanas de nível mais baixo devem ser satisfeitas antes das necessidades de nível mais alto. Dentre as necessitadas de baixo nível (necessidades fisiológicas, mais básicas) podemos citar, dentre outras, a necessidade de alimentação. Restaurantes são estabelecimentos que há vários séculos vêm suprindo essa necessidade, servindo refeições para as pessoas em troca de um pagamento.

Assim como qualquer outro estabelecimento comercial, os restaurantes buscam o lucro, o qual é proveniente dos serviços que prestam. A variedade de restaurantes existentes nas mais diversas cidades torna este um mercado extremamente competitivo. Para obter um lucro satisfatório é necessária uma quantidade razoável de consumidores. Segundo Chiavenato (2000), a conquista do cliente vai desde saber como abordá-lo, até receber as críticas com naturalidade, pois elas ajudam a melhorar o atendimento, sendo de suma importância para que a organização se torne competitiva no mercado com qualidade aprovada pelo cliente. Nesse sentido, a satisfação do cliente é um aspecto fundamental e que jamais deve ser ignorado.

A qualidade da comida, a localização do estabelecimento e o preço são fatores decisivos para conquistar muitos clientes e que podem fazer um restaurante se destacar menos ou mais do que outro, porém alguns restaurantes já se valem de outros métodos para se destacarem dos demais. Muitas empresas (nas mais diversas áreas) que fizeram diferente do convencional, inovando e surpreendendo o cliente, conquistaram várias pessoas e obtiveram enorme sucesso. No ramo da tecnologia temos como exemplos a Apple e a Microsoft. Dessa forma, certos restaurantes buscam meios diferentes dos usuais para atrair a atenção do cliente e conquistá-lo.

Os restaurantes tradicionalmente fazem o atendimento ao cliente por meio de um funcionário, o qual é chamado de garçom. Ele é o responsável por anotar os pedidos, comunicar os mesmos aos cozinheiros e, depois de prontos, trazê-los aos clientes. Essa forma de atendimento perdura há muitos anos, porém com o aumento na quantidade de consumidores (e consequentemente da demanda) nota-se que alguns problemas surgem.

Primeiramente, a agilidade e a qualidade no atendimento ao cliente ficam comprometidas. Isso porque quanto maior a quantidade de clientes aguardando ao mesmo tempo por atendimento, menor é a capacidade do restaurante de atendê-los de maneira ágil, pois a quantidade de garçons é limitada. Aumentar o número de garçons neste caso seria uma possível solução, contudo esse método acarretaria aumento na despesa do estabelecimento com funcionários. Feita esta observação, percebe-se que o restaurante apenas estaria substituindo um problema por outro.

Outro fator que merece destaque, mas que não ocorre comumente, é a falha humana. Um erro de comunicação pode fazer com que um pedido vindo de um cliente seja mal interpretado pelo garçom, o que faria o pedido errado ser entregue. Ou, ainda, pode ocorrer de o garçom anotar o pedido certo, todavia se confundir e entregá-lo na mesa errada.

Uma preocupação que os donos de restaurante têm hoje também é a atualização dos cardápios de papel. Ao fazer uma alteração no preço, adicionar ou retirar algum prato, entre outras mudanças, percebe-se que é necessário produzir novos cardápios. O problema aumenta conforme cresce a quantidade física de cardápios usada pelo estabelecimento. Se estes mesmos cardápios pudessem ser produzidos e disponibilizados para o cliente em um meio digital, seria possível realizar alterações como estas em poucos segundos, ao mesmo tempo em que se economizam recursos financeiros e papel.

Além destes e outros fatores, é preciso observar que a tecnologia já faz parte da vida de milhões de pessoas, facilitando os mais variados serviços e automatizando processos. Isso porque os sistemas tecnológicos estão menos suscetíveis a erro e proporcionam economia de

tempo para as empresas, pois realizam tarefas de maneira mais célere do que uma pessoa. Esse foi um fator determinante para que muitos restaurantes começassem a implementar alguma forma de automação tecnológica no processo de atendimento ao cliente. Ademais, os benefícios trazidos por esta implementação reduzem, de médio a longo prazo, muitos dos custos que os restaurantes possuem ao longo do ano, compensando o alto custo que se pode ter inicialmente, dado que muitas das soluções tecnológicas não são baratas de serem produzidas.

O restaurante Mangai (especializado em comida típica nordestina) possui unidades em Brasília, João Pessoa e Natal e tem investido cada vez mais em tecnologias que garantem uma melhor qualidade no atendimento e processos. Para facilitar a gestão dos restaurantes, o Mangai criou uma organização dedicada à criação e desenvolvimento de soluções para automação comercial gastronômica. O garçom Maicon de Azevedo Lima cita que a automação utilizada pelo Mangai torna o atendimento mais ágil. Por meio de um dispositivo móvel, ele envia o pedido à cozinha e já o adiciona à conta, precisando apenas digitar o número do cartão (que armazena os pedidos do cliente). A figura 2.1 mostra um garçom do Mangai utilizando um dispositivo móvel para anotar o pedido do cliente e adicioná-lo à conta.

Figura 1.1 – Dispositivo móvel para atendimento ao cliente do restaurante Mangai
(Fonte: http://www.mangai.com.br/site/qualidade/tecnologia/)

A rede Mangai aderiu a dois softwares:

- “Software de atendimento ao cliente – O garçom, portando um dispositivo móvel, através da tecnologia wireless, faz o pedido ao lado do cliente, que simultaneamente é enviado para impressoras na cozinha, eliminando, assim, o deslocamento do garçom até a área de produção, adquirindo com isso benefícios como: agilidade na entrega do

pedido, controle de vendas, agilidade e qualidade no atendimento e maior produtividade e venda.”

- “Software *admin*istrativo e financeiro – Programa que gerencia todos os fluxos de processos e informações da empresa, atendendo às áreas importantes: - gestão de estoque, vendas, compras de mercadorias, controle de produção, gestão financeira e gestão de pessoal.”

Outra solução tecnológica que vêm sendo adotada por alguns estabelecimentos como, por exemplo, o Outback (restaurante especializado em pratos da culinária australiana), é o Sistema de Longo Alcance (do inglês Long Range System – LRS). A figura 2.2 mostra um tipo de dispositivo LRS utilizado pelo Outback. Este dispositivo substitui o visor digital que é empregado nas situações nas quais o cliente deve aguardar para pegar o seu pedido em um balcão. Nestas situações, o cliente usualmente recebe uma senha e acompanha a sua vez pelo citado visor. Ocorre que, desse modo, o cliente é obrigado a virar a cabeça constantemente para visualizar a senha que está sendo chamada e assim não perder a sua vez.

Figura 1.2 – Sistema de Longo Alcance

(Fonte: http://www.sebraemercados.com.br/inovacao-no-atendimento-o-chamado-mais-personalizado/)

O LRS alerta o usuário sobre a chegada do seu pedido de maneira mais eficiente, oferecendo mais conforto e comodidade, além de reduzir a ansiedade. Funciona da seguinte

maneira: Ao fazer o pedido o cliente recebe o dispositivo LRS e, quando o pedido está pronto, o sistema vibra e acende luzes vermelhas.

Após os exemplos citados é fácil perceber que a automação traz uma série de benefícios e já é realidade em alguns estabelecimentos. O que determina a viabilidade ou não do emprego das soluções tecnológicas desenvolvidas são as características de cada restaurante (o modo como é feito o atendimento, etc.). É preciso um estudo detalhado de cada uma a fim de se tirar o máximo de proveito das vantagens que elas oferecem.

A solução proposta nesta monografia pretende diminuir o tempo de espera do cliente para fazer um pedido, além de fornecer ao cliente uma interface para tirar eventuais dúvidas e solicitar o fechamento da conta. Pretende-se, também, reduzir a quantidade de cardápios físicos utilizados, vistos os problemas que isto acarreta. O projeto pode ser aplicado em qualquer restaurante à la carte, porém é recomendado o seu uso apenas em restaurantes que recebem uma grande quantidade de clientes, pois são eles os que mais sofrem desses problemas.

1.2 – Objetivos

Este projeto tem por objetivo geral desenvolver um sistema para automatizar os pedidos em restaurantes por meio de um *tablet*, não sendo necessário chamar o garçom sempre que o cliente desejar fazer um novo pedido.

Os objetivos específicos são:

- Desenvolver uma interface por onde o cliente irá visualizar o cardápio e enviar seus pedidos e dúvidas.
- Desenvolver uma interface por onde a cozinha visualizará os pedidos e o caixa visualizará as contas que estão aguardando fechamento.
- Implementar um alerta sonoro com o auxílio de um microcontrolador Arduino, que irá soar um *buzzer* sempre que um novo pedido for enviado, alertando sobre a chegada do mesmo.
- Implementar um banco de dados relacional que armazenará os dados do usuário, cardápio, pedidos, entre outros.

1.3 – Justificativa e Importância do Trabalho

As empresas modernas têm, cada vez mais, buscado o apoio que a tecnologia da informação oferece ao mercado, ora facilitando processos, ora automatizando. Esta é uma realidade presente em vários setores, não importando se se trata de uma micro ou macro empresa. Fazer uso de ferramentas que propiciem a melhor experiência para os clientes não apenas traz vantagens competitivas para a empresa/estabelecimento como também podem se tornar um bom investimento a médio ou longo prazo.

Esforços já têm sido feitos para otimizar o atendimento ao cliente. Há locais, por exemplo, onde o garçom faz uso de dispositivos eletrônicos para anotar e comunicar o pedido aos cozinheiros, ao invés do tradicional bloco de papel e caneta. Porém mesmo nestes casos o cliente ainda necessita de um garçom disponível sempre que desejar fazer um novo pedido. Faz-se então necessária a adoção de outros métodos que possam melhorar a maneira como os clientes realizam seus pedidos, tornando-se um diferencial que trará melhorias aos negócios deste setor.

1.4 – Escopo do Trabalho

Será desenvolvida, em linguagem de programação PHP, uma aplicação web, que o usuário acessará por meio de um *tablet* (a ser fornecido pelo próprio estabelecimento, conforme solução aqui proposta). Por meio desta aplicação o usuário terá acesso ao cardápio e poderá fazer os pedidos, os quais o sistema se encarregará de enviar diretamente para a cozinha. Será também implementada uma solução em hardware utilizando um microcontrolador Arduino Uno, que irá acionar um *buzzer* e ligar um LED, alertando os responsáveis na cozinha sobre um novo pedido feito por um cliente.

O servidor web utilizado é o apache, que fornece todo o suporte necessário para o correto funcionamento do sistema. Os dados que alimentarão a aplicação são armazenados em um banco de dados MySQL. Ao enviar o pedido, o sistema guarda todas as informações necessárias para o atendimento, como qual foi o pedido, quantidade, observações do pedido, etc.

O projeto não contempla o modo de pagamento das contas, ficando a critério do estabelecimento. O sistema apenas auxilia o usuário dando a opção de informar o seu desejo em fechar a conta, notificando o caixa para que faça a devida cobrança.

1.5 – Resultados Esperados

Com a implementação deste projeto espera-se diminuir a dependência constante que o cliente possui em relação ao garçom, fazendo-o se sentir mais à vontade e reduzindo o tempo de espera da comunicação dos pedidos.

1.6 – Estrutura do Trabalho

Este trabalho está dividido em seis capítulos, dispostos da seguinte maneira:

- CAPÍTULO 1 – INTRODUÇÃO: Apresenta os principais aspectos a serem desenvolvidos no trabalho. Inclui qual o problema a ser resolvido, objetivos, justificativa e importância da busca por uma solução, escopo e resultados esperados.
- CAPÍTULO 2 – BASES METODOLÓGICAS PARA RESOLUÇÃO DO PROBLEMA: Descreve as metodologias, técnicas e ferramentas que foram utilizadas para a resolução do problema, além de justificar a escolha de cada uma delas.
- CAPÍTULO 3 – MODELO DE SOLUÇÃO PROPOSTO: Descreve a proposta do projeto para solucionar o problema, apresentação geral do modelo, descrição das etapas da implementação e suas contribuições.
- CAPÍTULO 4 – APLICAÇÃO PRÁTICA DO MODELO DE SOLUÇÃO: Apresenta uma análise sobre os locais onde o projeto pode ser melhor aplicado, resultados da implementação, custos para o desenvolvimento da solução e dificuldades encontradas.
- CAPÍTULO 5 – CONCLUSÃO: Traz uma reflexão acerca do alcance dos objetivos. Ressalta ainda as vantagens e limitações do projeto, além de sugestões para trabalhos futuros.

CAPÍTULO 2 – BASES METODOLÓGICAS PARA RESOLUÇÃO DO PROBLEMA

Este capítulo apresenta as principais técnicas, métodos, metodologias e outras ferramentas que foram utilizadas na solução do problema, além das justificativas que levaram à escolha de cada um (a).

2.1 – Microcontrolador

Microcontroladores são computadores (possuem processador, memória e periféricos de entrada e saída) que são "embutidos" dentro de algum outro dispositivo, com o intuito de controlar as funções ou ações de um produto. A principal diferença entre um computador de mesa (desktop) e um microcontrolador é que este é feito para realizar uma tarefa específica, enquanto o computador de mesa apresenta propósitos gerais (pode ser usado para executar várias tarefas, ao invés de uma tarefa pré-definida). Os microcontroladores normalmente possuem pequenas dimensões (cabendo na palma da mão) e consomem pouca energia para manter o seu funcionamento.

2.1.1 – Arduino

O microcontrolador Arduino apresentou-se como um dispositivo alternativo aos microcontroladores da época em que foi criado (que eram relativamente caros e difíceis de serem utilizados). O Arduino foi construído para ser um microcontrolador barato e de fácil utilização, o que fez com que sua popularidade crescesse. As placas do Arduino têm catorze pinos digitais, que podem ser definidos individualmente como entrada ou saída, e seis entradas analógicas. O Arduino pode ser usado em diversos tipos de projetos e vêm recebendo melhorias e novas funcionalidades, apresentando diversas versões atualmente (MCROBERTS, 2011).

A versão denominada "Uno" do Arduino será a empregada nesta solução. Dentre as facilidades encontradas neste microcontrolador podemos citar a alimentação do mesmo (que pode ser realizada por uma simples conexão USB), a comunicação simplificada com um computador ou outros microcontroladores e a quantidade de entradas suficiente para o propósito com o qual foi empregado (a ser descrito mais adiante). A figura 3.1 mostra um microcontrolador Arduino Uno.

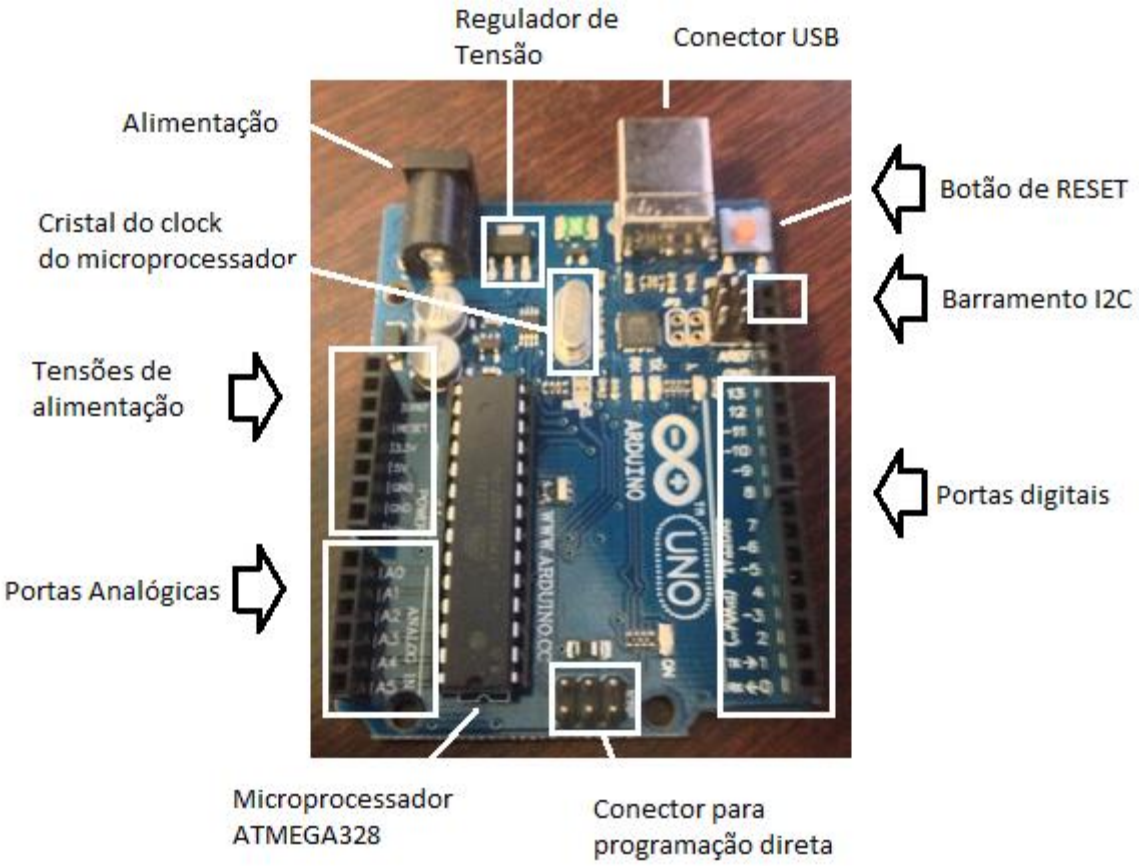

Figura 2.1 – Microcontrolador Arduino Uno

(Fonte: http://www.Arduinodolito.com.br/hardware-do-Arduino-uno/)

Neste projeto o Arduino será utilizado para acionar um *buzzer*, que soará durante três segundos, e acender um *led*, que permanecerá ligado até o responsável na cozinha pressionar um botão. Para maiores detalhes sobre o Arduino, consultar (MCROBERTS, 2011).

2.2 – Linguagens

2.2.1 – HTML

Uma das linguagens largamente utilizadas neste projeto é a linguagem "*Hyper Text Markup Language"* (HTML), que significa "Linguagem de Marcação de Hipertexto". O HTML é uma linguagem que está presente nas páginas web, e tem como principal funcionalidade definir a estrutura da página (indicar títulos, parágrafos, imagens, entre outros). O HTML consegue fazer isso por meio das denominadas *tags* (etiquetas), que são palavras colocadas entre os sinais de menor e maior. As *tags* informam ao navegador a

estrutura e o significado do seu texto. Dessa forma, a *tag* "<p>", por exemplo, indica que aquele conteúdo se trata de um parágrafo (FREEMAN, 2008). O lado ruim do HTML é que ele não oferece muitos meios para o usuário interagir com a página dinamicamente, como em uma página de *login*, onde se espera que um usuário digite uma senha e, caso esta senha não esteja correta, o sistema retorne um aviso informando que o dado informado é inválido. Neste caso é necessário fazer uso de linguagens adicionais que trabalhem em conjunto com o HTML, tal como o *JavaScript*. Para maiores detalhes sobre o HTML, consultar (FREEMAN, 2008).

2.2.2 – *JavaScript*

O *JavaScript* é uma linguagem de programação, que especifica o comportamento de uma página web. Esta linguagem é capaz de modificar o HTML de forma dinâmica, de acordo com as ações do usuário do sistema. Assim, quando um usuário move o *mouse* em cima de um elemento ou clica em um botão, o *JavaScript* permite exibir uma mensagem ou mesmo mudar a posição de outros elementos na estrutura da página, entre várias outras possibilidades. Uma das principais vantagens do *JavaScript* é que este é executado no lado do cliente (mais especificamente no navegador web). Isso reduz consideravelmente a carga de processamento no servidor. Podemos citar como exemplo uma página de cadastro, na qual o usuário digita suas informações pessoais e as envia para o servidor. Sem o *JavaScript,* as informações seriam validadas apenas no servidor e, caso estas estivessem erradas ou com campos obrigatórios não preenchidos, o servidor não completaria o cadastro, sendo necessário que o usuário digitasse as informações novamente. O *JavaScript* permite fazer estes tipos de validações de dados no navegador, sem a interferência do servidor. As informações do formulário, deste modo, só são enviadas quando estão corretamente preenchidas, evitando processamento desnecessário no servidor.

Neste projeto o *JavaScript* será usado principalmente para fazer validações de formulários, além de mostrar e atualizar dados dinamicamente. Para maiores detalhes sobre o *JavaScript*, consultar (FLANAGAN, 2011).

2.2.3 – CSS

A principal linguagem adotada para criar o *design* de páginas *web* é a linguagem "*Cascading Style Sheets"* (CSS), que significa "Folhas de Estilo em Cascata". Enquanto o

HTML define a estrutura da página, o CSS define a aparência da mesma. Com o CSS é possível definir as cores de fundo da página, a fonte dos textos, tamanho de imagens, etc. O CSS possibilita definir um conjunto de regras de apresentação tanto para uma única página HTML específica quanto para várias páginas ao mesmo tempo. Essa característica evita que se escreva código duplicado, pois o código pode ser compartilhado.

O termo “Cascata” do CSS se refere a uma característica importantíssima, que é o critério de escolha de qual regra deve ser aplicada na página. Imagine, por exemplo, uma situação na qual temos duas regras que atingem uma mesma *tag* HTML. Nesse caso, o CSS adotará a regra que for mais específica. Desta maneira, se a primeira regra descreve que todas as *tags* “<p>” devem ser apresentadas em negrito e a segunda regra descreve que uma *tag* “<p>” em particular não deve estar em negrito, esta última será a regra a ser adotada para esta *tag* específica. Nota-se que as regras mais gerais só serão utilizadas quando não houver outra que seja mais específica. Para maiores detalhes sobre o CSS, consultar (FREEMAN, 2008).

2.2.4 – XML

A linguagem “*eXtensible Markup Language”* (XML), que significa “Linguagem Extensível de Marcação”, é uma linguagem usada para padronizar dados com o objetivo de organizar, separar o conteúdo e integrá-lo com outras linguagens. A principal vantagem do XML é que sua sintaxe permite que diferentes computadores e aplicações compartilhem dados e informações. Deste modo, os dados presentes em um banco de dados podem ser lidos por outro banco. Isso aumenta a portabilidade dos dados, que não necessitam de um processo mais trabalhoso para serem migrados. Neste projeto o XML será usado para trafegar informações nas requisições AJAX. Para maiores detalhes sobre o XML, consultar (BRAY, 2004).

2.2.5 - PHP

O PHP (originalmente um acrônimo para “Personal Home Page”) é uma linguagem de programação livre que atua no lado do servidor. O código do PHP é interpretado no servidor web e gera HTML como saída, o qual será por sua vez visualizado pelo usuário (no lado do cliente). Suas principais funcionalidades são de grande utilidade na web, o que fez com que fosse adotado em milhões de páginas de que se tem registro. Importante ressaltar que o PHP é uma linguagem de código aberto e pode ser usado, alterado e redistribuído sem precisar pagar

por isso.

As vantagens desta linguagem que levaram a escolha da mesma são várias. Dentre elas podemos destacar:

- Interfaces para vários bancos de dados diferentes
- Bibliotecas prontas para várias tarefas comuns na web
- Baixo custo
- Fácil de aprender e usar
- Portabilidade
- Disponibilidade do código fonte

O PHP também é capaz de abrir portas seriais e enviar dados através delas. Assim sendo, será utilizado na comunicação com o Arduino. Além disso o PHP será o principal responsável pela comunicação com o banco de dados, dentre várias outras funcionalidades. Para maiores detalhes sobre o PHP, consultar (WELLING, 2003).

2.3 – Biblioteca JQuery

JQuery é uma das várias bibliotecas do *JavaScript*, sendo a mais popular de todas elas. Através do *JQuery*, todas as funcionalidades e vantagens do *JavaScript* podem ser utilizadas, porém de maneira muito mais simples. O *JQuery* possui código aberto e pode ser alterado por qualquer pessoa, o que torna possível, por exemplo, a criação de *plugins* (programa criados para atender uma funcionalidade específica). Alguns *plugins JQuery* que já se encontram disponíveis são utilizados para fazer edições de imagens, validações de formulários, etc. Isso facilita bastante o trabalho do desenvolvedor e diminui o tempo necessário para a construção de uma aplicação web. Neste projeto o *JQuery* será utilizado para adicionar alguns efeitos visuais na tela de cardápio e implementar a comunicação AJAX de maneira mais simplificada. Para maiores detalhes sobre o *JQuery*, consultar (SILVA, 2008).

2.4 – AJAX

AJAX é um acrônimo em inglês para “Asynchronous Javascript and XML”, que significa “Javascript e XML Assíncronos”. AJAX se refere a um conjunto de técnicas que fazem uso de Javascript e XML para enviar e trazer informações de forma assíncrona. O AJAX funciona de maneira transparente e permitiu elevar o nível de interatividade entre o

usuário e a página web, pois possibilita carregar dados sem que para isso seja necessário recarregar a página inteira. Essa característica faz com que a navegação se torne mais rápida. Neste projeto o AJAX será utilizado para solicitar algumas operações específicas e retornar resultados de maneira assíncrona. Para maiores detalhes sobre o AJAX, consultar (GARRETT, 2005).

2.5 – Protocolos

Protocolos são regras e convenções usadas para se estabelecer uma comunicação entre dois ou mais dispositivos. Trata-se da "linguagem" através da qual os dispositivos "conversam", ou seja, trocam mensagens e dados. A violação do protocolo dificulta a comunicação, podendo até mesmo torná-la impossível. Um dos protocolos de fundamental importância para comunicação na web é o HTTP.

2.5.1 – HTTP

HTTP significa "Protocolo de Transferência de HiperTexto" (do inglês "*HyperText Transfer Protocol"*) e é o protocolo usado na comunicação entre clientes e servidores na *web*. Ele é responsável por especificar as mensagens que o cliente pode enviar ao servidor e quais as respostas a serem recebidas de volta. O cabeçalho do HTTP armazena informações como o tipo de dados que estão sendo transportados, identificação do servidor, entre outros. O protocolo HTTP será utilizado para trafegar dados e informações do servidor para o cliente e vice-versa. Para maiores detalhes sobre o HTTP, consultar (FIELDING, 1999).

2.6 – Servidor

Um servidor é um processo que fornece um serviço específico a um ou mais clientes. Quando um cliente solicita um serviço ao servidor, este processa a requisição e empacota os resultados em uma mensagem de resposta, que é enviada ao cliente. Um servidor web normalmente recebe e atende requisições de páginas web, se comunicando com o cliente por meio do protocolo HTTP.

2.6.1 – Apache

O servidor Apache é o servidor web mais popular e bem-sucedido do mundo, sendo que atualmente mais da metade dos sites existentes na web rodam em Apache. Dentre as vantagens do Apache que levaram a sua escolha podemos destacar o fato de possuir uma excelente performance, segurança e compatibilidade com diversas plataformas, além de ser um software livre (o que possibilita que ele seja melhorado por diversas pessoas com o passar dos anos). Para maiores detalhes sobre o Apache, consultar (ALECRIM, 2010).

2.7 – Cliente

Um cliente trata-se de um processo que requisita um serviço de um servidor. Para solicitar um serviço, o cliente manda uma mensagem ao servidor, contendo o endereço, serviço desejado, dados para serem processados (caso necessário), etc. Na web, o cliente usualmente refere-se ao *browser* (navegador), que solicita páginas e outros serviços de um servidor web. O cliente escolhido para este projeto foi o *browser* google chrome, porém o sistema também deve ser corretamente exibido em outros *browsers.*

2.8 – Arquitetura de Software

Segundo Pressman (2011), "A arquitetura de software de um programa ou sistema computacional é a estrutura ou estruturas do sistema, que abrange os componentes de software, as propriedades externamente visíveis desses componentes e as relações entre eles". A arquitetura nos permite compreender o sistema de maneira mais simples, separar os problemas e atender os requisitos de maneira mais eficiente.

2.8.1 – MVC

O model-view-controller (modelo-visão-controlador, em português) consiste em um modelo de arquitetura de software que foi criado pensando-se em dividir as partes e conceitos de um sistema, tornando-os mais fáceis de identificar, implementar, gerenciar e reusar. A separação de responsabilidades e o reuso foram fatores que fizeram com que a utilização deste modelo se desse principalmente em sistemas orientados a objeto.

O modelo consiste principalmente nos dados e no processamento e manipulação dos mesmos. A visão compreende a parte da aplicação com a qual o usuário tem contato, se tratando de uma interface gráfica em boa parte dos casos. O controlador é responsável por

fazer a “intermediação” entre a visão e o modelo, normalmente recebendo requisições da visão e as repassando para o modelo, responsável por tratá-las. Para maiores detalhes sobre o MVC, consultar (DEACON, 2009).

2.9 – Banco de Dados

Um banco de dados é uma estrutura formada por uma coleção de dados (geralmente de usuários finais de sistemas) e metadados (descrições das características dos dados e do conjunto de relacionamentos que ligam esses dados). Os bancos de dados normalmente são gerenciados e controlados por um Sistema de Gerenciamento de Bancos de Dados (SGBD). O SGBD é formado por um conjunto de programas que, entre outras funções, controlam os acessos aos dados armazenados e gerenciam a estrutura do banco.

2.9.1 – MySQL

O MySQL é um SGBD rápido e robusto que permite armazenar, buscar e recuperar dados. Permite o acesso de múltiplos usuários, porém restringindo esse acesso por meio de permissões para que somente a pessoa devidamente autorizada possa realizar procedimentos no banco. O MySQL usa SQL (acrônimo de “Structured Query Language”), que significa “Linguagem de consulta estruturada”. O SQL é uma linguagem através da qual podemos fazer operações no banco de dados (seja consulta de dados, escrita de dados, alterações de dados, entre outras). O MySQL será utilizado para armazenar as informações principais do sistema, como itens do cardápio, pedidos e avaliações dos clientes, entre outras. Para maiores detalhes sobre o MySQL, consultar (WELLING, 2003).

2.10 – Componentes Físicos

2.10.1 – *Buzzer*

O *buzzer* é um dispositivo utilizado para emitir bipes sonoros. Apresenta as vantagens de ser pequeno, consumir pouca energia e ter baixo custo. O *buzzer* será utilizado para emitir som sempre que um cliente solicitar um pedido.

2.10.2 – LED

LED é a sigla para Light Emitting Diode, que em tradução livre significa diodo emissor de luz. O LED é um componente eletrônico semicondutor que tem a capacidade de transformar energia elétrica em luz. O LED será utilizado para, juntamente com o *buzzer*, alertar sobre uma nova solicitação de pedido do cliente.

2.11 – *Tablet*

O *tablet* é uma espécie de computador portátil de dimensões relativamente pequenas. Usualmente possui tela sensível ao toque. É muito utilizado para acessar páginas web, visualizar imagens e realizar leituras de livros e periódicos, sendo que não necessita de dispositivos físicos como *mouse* e teclado. Sua bateria possui boa autonomia se comparada com a de outros computadores portáteis. A maior vantagem de um *tablet* é ser um dispositivo simples e rápido para a visualização de conteúdos. O *tablet* foi escolhido neste projeto para ser o dispositivo através do qual o cliente acessa o sistema.

O próximo capítulo discorrerá sobre o modelo proposto para solucionar o problema apresentado nos capítulos anteriores. Serão também detalhados os passos que devem ser seguidos para implementar a solução.

CAPÍTULO 3 – MODELO DE SOLUÇÃO PROPOSTO

Este capítulo tem por objetivo fornecer, inicialmente, uma visão geral do modelo proposto, fazendo-se em seguida uma descrição mais detalhada de cada etapa do modelo (objetivos da etapa, forma de implementá-la, entre outros). Para facilitar o entendimento serão utilizados, sempre que possível, recursos visuais tais como figuras e diagramas.

3.1 – Apresentação Geral do Modelo de Solução

O projeto consiste em um sistema web, acessível por qualquer *browser*, com quatro perfis (tipos) de usuário: *admin*, *cozinha*, *caixa* e *cliente*. Ao fazer o *login*, o sistema salva o perfil do usuário e redireciona o mesmo para a página inicial correspondente de acordo com o perfil. Importante ressaltar que em cada página solicitada é feita uma verificação para se ter certeza de que o usuário que está solicitando aquela página possui autorização para acessá-la. Caso o usuário não tenha autorização, ele será redirecionado para a página de *login*. O único perfil de usuário que possui permissão de acesso a qualquer página, sem restrições, é o *admin*. As funcionalidades do sistema disponíveis para cada perfil estão dispostas da seguinte maneira:

- O perfil *cliente* terá acesso ao cardápio digital, poderá realizar pedidos, enviar dúvidas, acompanhar o valor da conta, solicitar o fechamento da conta, entre outros.
- O perfil *cozinha* terá acesso à lista de pedidos e lista de dúvidas dos clientes, além de poder adicionar um novo prato ao cardápio.
- O perfil *caixa* terá acesso à lista de contas solicitadas pelo cliente.
- O perfil *admin* terá acesso à lista de avaliações dos clientes em relação ao atendimento. Este perfil também faz o gerenciamento dos usuários com perfil *caixa* e *cozinha*, sendo o único capaz de alterar, adicionar ou remover usuários com este perfil.

A figura 3.1 apresenta a visão geral da solução proposta. Para cada mesa do restaurante será disponibilizado um *tablet*, através do qual os clientes poderão acessar o sistema. Todas as mesas deverão estar devidamente identificadas e numeradas. O usuário então irá escolher um número de mesa ao fazer o *login* no sistema, de modo que os outros perfis saibam de qual mesa veio cada requisição. Os demais perfis (*caixa*, *cozinha* e *admin*) acessarão o sistema por meio de computadores de mesa. No momento em que o cliente envia um pedido ao usuário

cozinha, um sinal é enviado para o Arduino que estará conectado no computador acessado por este, acionando um *buzzer* e ligando um LED. Dessa forma, o funcionário na cozinha será alertado sobre a chegada de novos pedidos. O usuário *cliente* também poderá solicitar o fechamento da conta ao usuário *caixa* por meio do *tablet*. Por fim, o usuário *admin* faz o gerenciamento dos usuários *cozinha* e *caixa*.

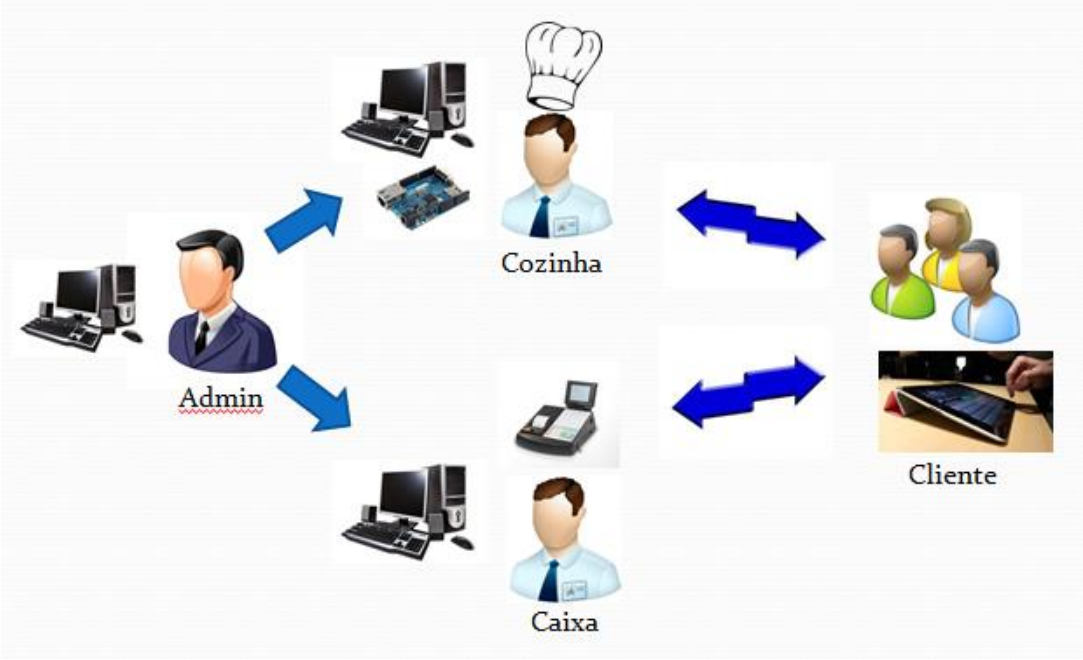

Figura 3.1 – Visão geral da solução proposta

(Fonte: Autor)

3.2 – Descrição da Solução

3.2.1 – Estruturação do sistema

A estrutura do sistema, seguindo o modelo MVC, foi dividida e organizada conforme mostra a figura 3.2.

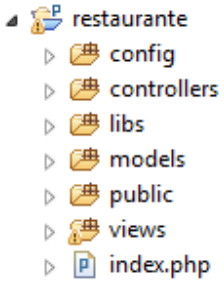

Figura 3.2 - Organização da estrutura do sistema

(Fonte: Autor)

Primeiramente temos um diretório (pasta) principal nomeado de “restaurante”, onde fica armazenado todo o código fonte do software. Dentro deste diretório raiz dividimos o código em subdiretórios, sendo que cada um deles é responsável por uma parte fundamental para o funcionamento do software.

No subdiretório “config” ficam armazenadas as constantes de configuração. Aqui foram definidas as informações para conexão com o banco de dados, como nome do banco, usuário e senha. Assim, em qualquer parte do sistema em que essas informações sejam necessárias é preciso apenas utilizar o nome da constante. Caso alguma informação mude futuramente, basta mudar a informação que a constante armazena. Dessa forma, se o nome do banco de dados muda, tudo que se precisa fazer é alterar o valor da constante que guarda o nome do banco, em vez de alterar o nome do banco em todos os locais do código que fazem uso dessa informação para se conectar.

O subdiretório “controllers” contém os controladores (*controllers*) de cada parte da aplicação. Os controladores têm por função principal invocar os métodos desejados dos modelos (*models*) de acordo com as requisições feitas pelo usuário.

O subdiretório “libs” possui os arquivos mais fundamentais da aplicação. Dentro dele existem arquivos que farão a conexão com o banco de dados, invocarão os controladores, modelos e visões (*views*) que serão utilizados de acordo com a URL da página e estabelecerão uma sessão para o usuário que realizou login no sistema.

O subdiretório “models” armazena os modelos do sistema. Os modelos recebem requisições dos controladores de acordo com as ações do usuário da aplicação, processam essas requisições e retornam um resultado. A maioria dos modelos recebem um ou mais dados de entrada e fazem operações a partir destes. Boa parte destas operações envolvem leitura, escrita ou alteração no banco de dados.

No subdiretório “public” ficam salvos os arquivos CSS, JavaScript e imagens de uso geral da aplicação. Estes arquivos serão compartilhados por várias partes do sistema, sendo aqui armazenados de modo a propiciar o reúso e evitar duplicidade de código.

O subdiretório “views” contém as visões, que são arquivos que fazem a interação direta com o usuário. As visões apresentam as interfaces e os dados ao usuário. É por meio da visão que o usuário solicita uma operação ao sistema.

O arquivo “index.php” é o único arquivo que se encontra diretamente logo abaixo do diretório raiz, sendo o responsável por iniciar toda a aplicação. Este arquivo faz uma requisição dos arquivos indispensáveis para o funcionamento do sistema (sendo a maior parte deles localizados no subdiretório “libs”) e os inicia.

3.2.2 – Banco de dados

Será aqui abordada a estrutura do banco de dados, fazendo-se uma descrição de cada tabela criada, sua importância e seus atributos (colunas). Ao final, serão mostrados os relacionamentos entre as tabelas.

3.2.2.1 – Estrutura do banco de dados

Para este projeto foi criado um banco de dados MySQL com seis tabelas, sendo elas: usuarios, contas, duvidas, cardapio, pedidos e avaliacoes. O banco foi nomeado de "projeto_restaurante". A figura 3.3 mostra um Diagrama Entidade Relacionamento do banco de dados (DER).

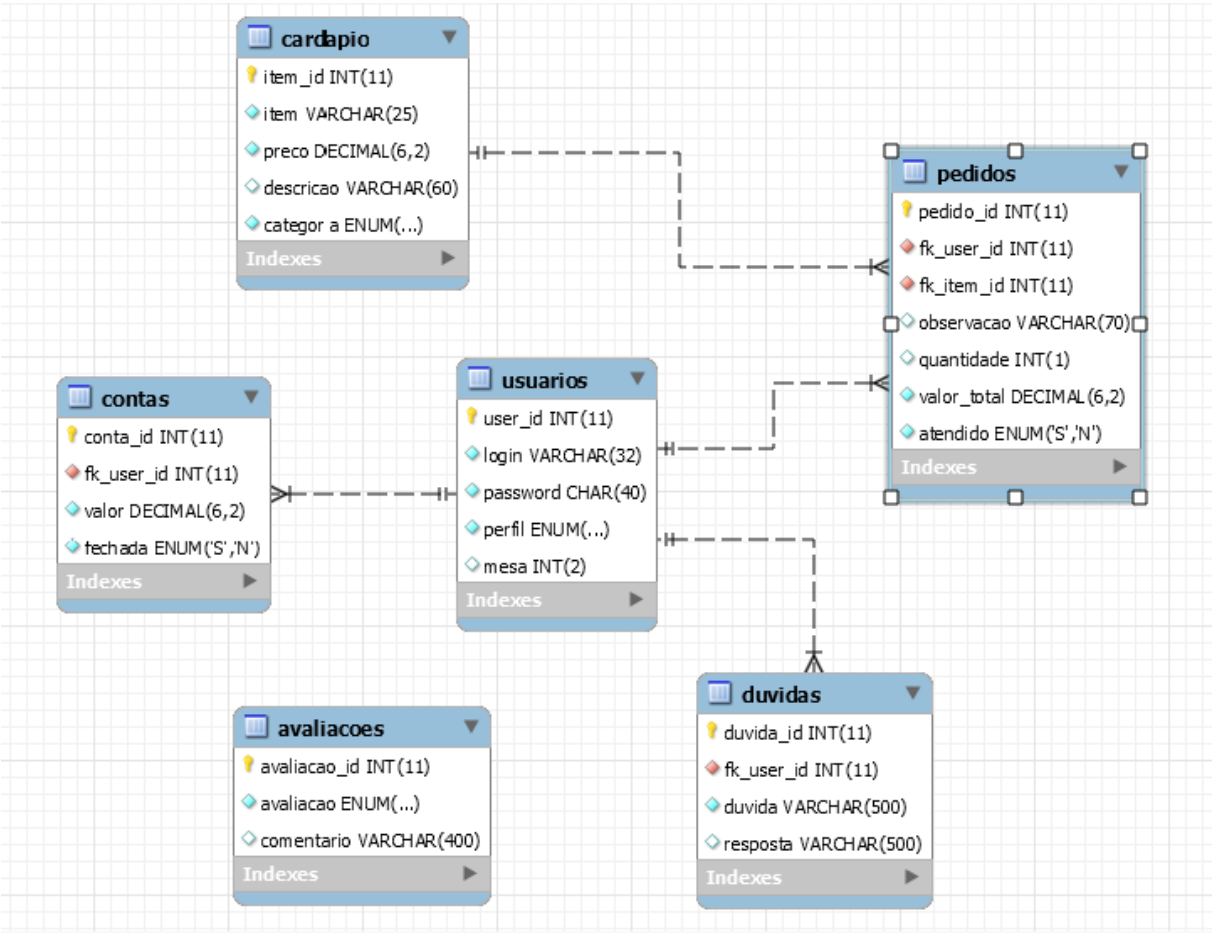

Figura 3.3 – Diagrama Entidade Relacionamento do banco de dados

(Fonte: Autor)

A tabela usuarios contém os dados dos usuários do sistema. Primeiramente temos a

coluna user_id, que é a chave primária da tabela. A coluna login representa o endereço de e-mail do usuário com perfil *cliente* e qualquer conjunto de caracteres para os demais perfis. Esta informação (juntamente com a senha) será necessária para o usuário fazer *login* no sistema. A coluna password se trata da senha do usuário. Como as senhas são dados que necessitam sigilo, estas são salvas apenas após passarem por uma função *hash* SHA-1. A coluna perfil armazena o perfil do usuário. Esta coluna aceita apenas quatro valores: *admin*, *caixa*, *cozinha* e *cliente*. A última coluna desta tabela é a coluna mesa, que salva o número da mesa escolhida pelo usuário com perfil *cliente*. Esta coluna sempre terá valor nulo (NULL) para os perfis *admin*, *caixa* e *cozinha*.

A tabela pedidos contém as informações dos pedidos dos clientes. Essas informações serão posteriormente visualizadas pelo usuário com perfil *cozinha*. A coluna pedido_id dessa tabela é a chave primária da mesma. A coluna fk_user_id é a chave estrangeira, que salva o número da chave primária do usuário que solicitou o pedido. A coluna fk_item_id se trata de outra chave estrangeira, que salva o número da chave primária do item que o usuário solicitou. A coluna observacao salva quaisquer observações que o usuário tenha feito no pedido. A coluna quantidade salva a quantidade do item que o cliente pediu. A coluna valor_total contém o valor total do pedido. O valor total é calculado a partir do produto entre o preço unitário do item e a quantidade solicitada. Por fim, a coluna atendido salva o "*status"* do pedido, isto é, se o pedido solicitado já foi ou não atendido. Esta coluna aceita somente dois valores: "S" (sim) ou "N" (não).

A tabela cardapio contém os itens que podem ser pedidos pelo usuário. A primeira coluna, item_id, é a chave primária desta tabela. A coluna item contém o nome do item do cardápio. A coluna preco registra o preço unitário do item. A coluna descricao contém quaisquer detalhamentos acerca do item que o restaurante queira fazer para o cliente. Esta coluna não é de preenchimento obrigatório. A última coluna, categoria, representa a categoria do item. Para este projeto foram criados quatro tipos de categorias, sendo eles: bebida, prato, acompanhamento e sobremesa.

A tabela duvidas tem por finalidade registrar as dúvidas do cliente. A coluna duvida_id é a chave primária. A coluna fk_user_id é a chave estrangeira que armazena a chave primária do usuário que enviou a dúvida. A coluna duvida se trata da dúvida em si. Já a coluna resposta guarda a resposta à dúvida do cliente.

A tabela contas mantém o registro de todas as solicitações de fechamento de conta. A coluna conta_id trata-se da chave primária da tabela. A coluna fk_user_id é a chave estrangeira que armazena a chave primária do cliente que solicitou o fechamento da conta. A coluna valor armazena o valor da conta a ser fechada. A coluna fechada salva o "*status"* da conta, isto é, se a conta solicitada já foi ou não atendida (fechada). Esta coluna aceita somente dois valores: "S" (sim) ou "N" (não).

A última tabela, avaliacoes, é usada para registrar as avaliações do usuário em relação ao atendimento. A coluna avaliacao_id é a chave primária da tabela. A coluna avaliacao representa a avaliação feita pelo usuário. Esta coluna aceita somente os seguintes valores: péssimo, fraco, regular, bom e ótimo. Por último temos a coluna comentario, a qual contém quaisquer comentários do porquê o usuário avaliou o atendimento daquela maneira. Esta coluna não é de preenchimento obrigatório.

3.2.2.2 – Relacionamento entre tabelas

Quanto aos relacionamentos entre as tabelas, temos que a tabela usuarios possui relacionamento do tipo Um para Muitos com a tabela pedidos, Um para Muitos com a tabela duvidas e Um para Muitos com a tabela contas. Apesar da tabela contas só poder ter uma conta em aberto para cada usuário, seu tipo de relacionamento (Um para Muitos) com a tabela usuarios se justifica pelo fato de que cada usuário pode possuir várias contas que já foram fechadas, pois os dados desta tabela não serão deletados. A tabela cardapio possui relacionamento do tipo Um para muitos com a tabela pedidos. Por último, a tabela avaliacoes é a única tabela que não possui relacionamentos com nenhuma outra tabela. Esta tabela poderia até possuir um relacionamento com a tabela usuarios, contudo optou-se por não relacionar essas tabelas para não identificar o cliente que fez a avaliação.

3.2.3 – Implementação do *hardware*

Assim que o cliente envia um pedido, uma requisição é feita ao servidor, que processa um código PHP responsável por inserir as informações do pedido no banco de dados. Logo em seguida o PHP abre uma conexão com o Arduino na porta serial onde este está conectado (normalmente é a porta COM4). O Arduino estará localizado na cozinha. A função que abre a

conexão com a porta é “$port = fopen('COM4', 'w')”. O primeiro parâmetro se trata da porta na qual se dará a comunicação entre o programa e o Arduino. Já o segundo parâmetro informa o tipo de operação que será realizada, que neste caso é uma operação de escrita (“W”). Em seguida uma *string* (“ligar”) é enviada através da porta serial para o *buffer* do Arduino, que por sua vez, ao verificar a existência desta *string*, ativa um buzzer durante três segundos e liga um LED, alertando a cozinha sobre a chegada do novo pedido. O processo descrito neste parágrafo irá se repetir sempre que algum cliente enviar um novo pedido.

A figura 3.4 apresenta um esquemático do circuito que alerta sobre a chegada de novos pedidos. Ao ser acionado pelo Arduino, o LED permanecerá ligado até que o responsável pela cozinha aperte um botão (*push button)*. O botão, por sua vez, enviará um sinal para que o Arduino desligue o LED. Foram adicionados também dois resistores, sendo um de 330 ohms (conectado ao LED) e outro de 8200 ohms (conectado ao botão).

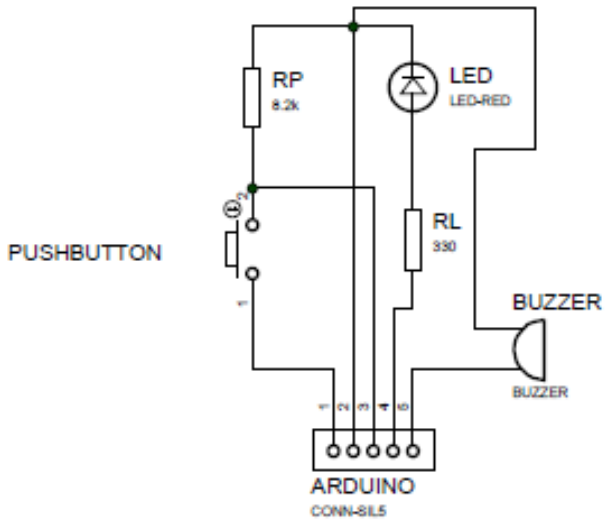

Figura 3.4 – Esquemático do circuito de alerta de pedidos

(Fonte: Autor)

A figura 3.5 mostra a ligação entre o Arduino e os componentes no *protoboard*. Esta figura mostra o estado dos componentes físicos após o envio de um pedido. Note que o LED encontra-se ligado, indicando que há um ou mais pedidos novos aguardando atendimento. Os itens numerados na figura representam, respectivamente:

1. LED;
2. Botão (*push button*);

3. *Buzzer.*

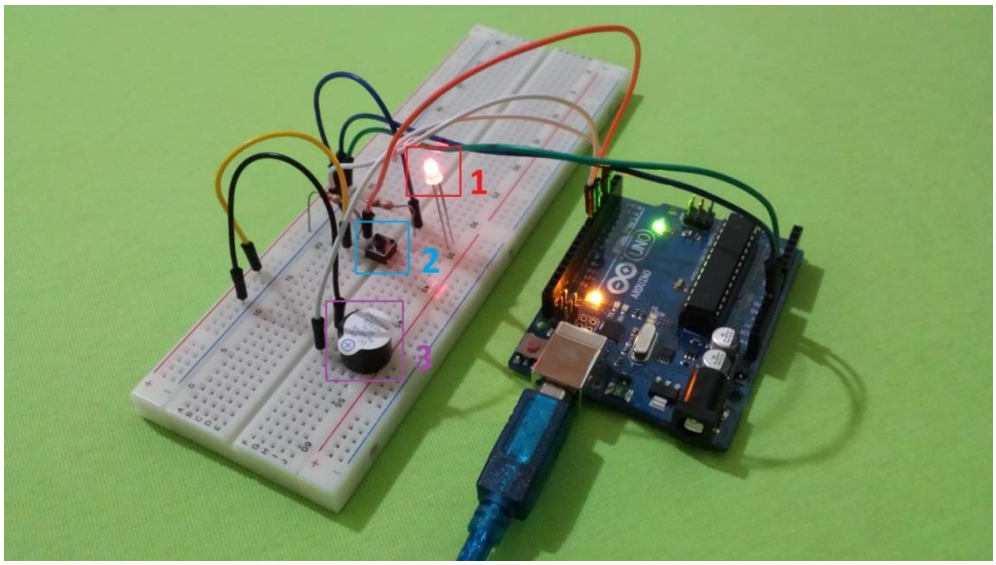

Figura 3.5 – Ligação entre o Arduino e os componentes

(Fonte: Autor)

3.2.4 – Página de *login*

A página de *login* é a única página de visibilidade pública, isto é, que pode ser acessada e visualizada por qualquer usuário. A figura 3.6 mostra a tela de *login* da aplicação.

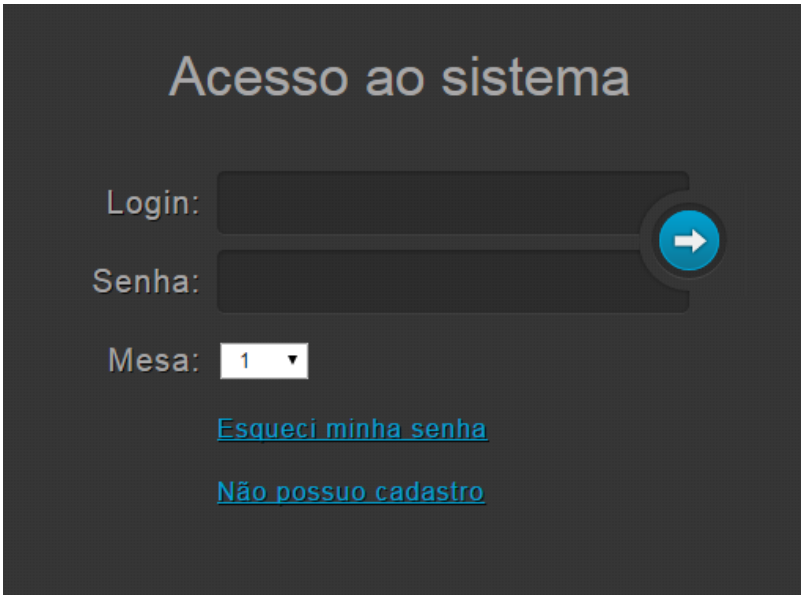

Figura 3.6 – Página de *login*

(Fonte: Autor)

Para ter acesso ao sistema, o usuário deverá preencher os campos *login* e senha e escolher uma mesa. Este último passo (escolher mesa) só é necessário para o usuário com perfil *cliente*, pois, no momento que faz a validação do *login* e senha, o sistema verifica também para este tipo de perfil se a mesa escolhida já está sendo utilizada por algum usuário. Se a mesa estiver disponível, o sistema concede o acesso à aplicação e insere no banco de dados o número da mesa que aquele usuário está utilizando. Caso contrário o sistema retorna uma mensagem informando que a mesa não está disponível e solicita que o usuário escolha outra. Se o usuário que estiver solicitando acesso tiver perfil *admin*, *cozinha* ou *caixa* o sistema ignora o campo mesa do formulário de *login* no momento da validação.

Após inserir corretamente o *login* e senha e enviar esses dados, o sistema redireciona o usuário da aplicação para sua página padrão de acordo com seu perfil. Essa página será comentada em detalhes mais adiante.

Na tela de *login* estão também disponíveis dois *links*, com as descrições "Esqueci minha senha" e "Não possuo cadastro". O primeiro *link* é utilizado para que o usuário possa recuperar a senha, caso tenha esquecido. Ao clicar neste *link* é mostrado ao usuário um formulário para que ele digite o e-mail que cadastrou no sistema, conforme mostra a figura 3.7. Após o usuário digitar o e-mail e clicar em "Enviar", o sistema verifica se aquele e-mail está cadastrado no banco de dados. Caso a resposta seja afirmativa a aplicação gera uma nova senha aleatória de oito caracteres e a envia para o e-mail informado. Caso contrário uma mensagem é exibida informando que aquele e-mail não está cadastrado. Essa nova senha será utilizada pelo usuário para ter acesso ao sistema. Posteriormente será mostrado como o usuário pode alterar esta senha.

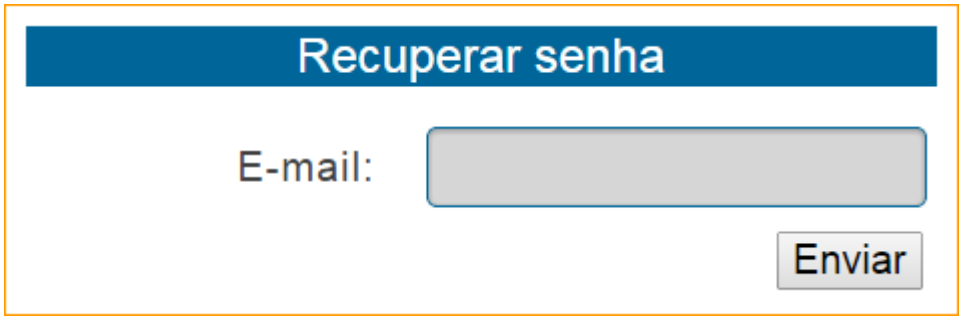

Figura 3.7 – Formulário de recuperação de senha

(Fonte: Autor)

O segundo *link* é utilizado para que os usuários ainda sem cadastro possam se cadastrar. Ao clicar neste *link* é mostrado ao usuário um formulário para que ele digite os

dados necessários para o cadastro no sistema, conforme mostra a figura 3.8. Os dados solicitados são: e-mail (representa o *login* do usuário com perfil *cliente*), senha e confirmação de senha. A senha não pode possuir menos do que seis caracteres. Após o usuário digitar os dados e clicar em "Enviar", o sistema verifica se o endereço de e-mail digitado é válido e se o mesmo ainda não está cadastrado no sistema, além de verificar também se as senhas digitadas coincidem e possuem o número mínimo de caracteres permitidos. Caso os dados estejam de acordo com as regras, o sistema insere o novo usuário no banco de dados. Usuários cadastrados por meio deste formulário por padrão serão usuários com perfil *cliente*. Somente o usuário com perfil *admin*, em sua página protegida, pode cadastrar usuários com perfil *cozinha* e *caixa*.

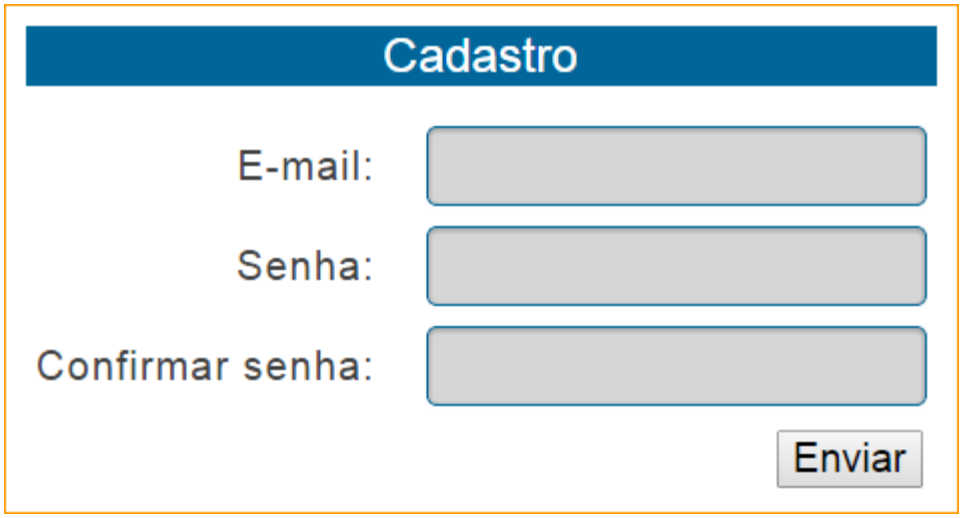

Figura 3.8 – Formulário de cadastro

(Fonte: Autor)

As operações de envio de nova senha ao e-mail e cadastramento de novo usuário, citadas nos formulários anteriores, são requisitadas de maneira assíncrona via *JQuery* e AJAX. O PHP as processa e devolve um resultado, sem necessidade de recarregar a página inteira.

3.2.5 – Interface que implementa o perfil *cliente*

Para acessar o sistema, o usuário com perfil *cliente* informa seu *login* (que no caso deste usuário se trata do e-mail digitado no cadastro) e senha. A página padrão para o qual este usuário é redirecionado após o login é a página de cardápio, mostrada na figura 3.9. O

usuário com perfil *cliente* possui um menu com acesso a três páginas, sendo elas: cardápio, dúvidas e alterar senha.

Figura 3.9 – Página de cardápio

(Fonte: Autor)

A página de cardápio é a principal página a ser utilizada pelo cliente. Nesta página o usuário visualiza todos os itens do cardápio, os pedidos realizados e acompanha o valor da conta. Os itens do cardápio estão separados por categoria. Para realizar um pedido é necessário que o cliente toque o dedo no item desejado. Ao fazer isso é aberto um formulário (figura 3.10), no qual o usuário pode escolher a quantidade que deseja daquele item e adicionar eventuais observações (opcionais). Um exemplo de observação seria “Retirar a cenoura”. Após isso basta tocar em “Adicionar” para que seu pedido seja enviado. Antes que o pedido seja inserido no banco de dados é feita uma validação prévia, que tem por finalidade verificar se o usuário já solicitou o fechamento da conta. Caso a resposta seja afirmativa, o sistema impede a inserção do pedido no banco e exibe uma mensagem ao usuário informando que a operação não foi possível porque a solicitação do fechamento da conta já foi feita.

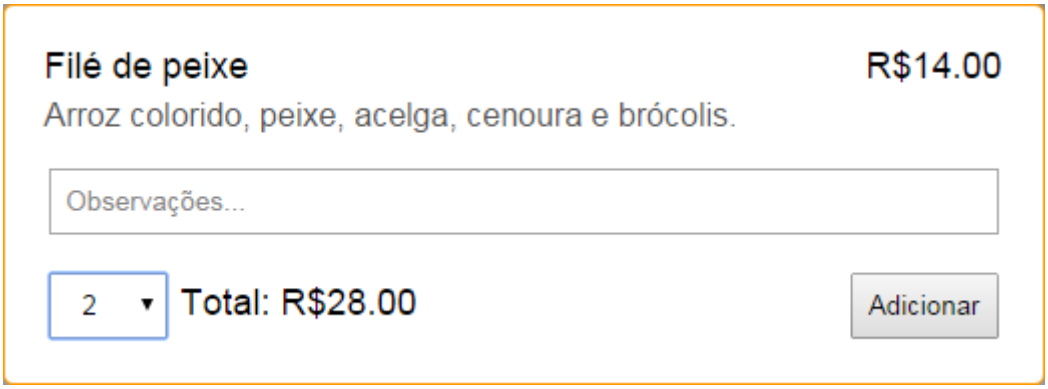

Figura 3.10 – Formulário de detalhamento do pedido

(Fonte: Autor)

Os pedidos vão sendo mostrados na coluna “Meus pedidos” (figura 3.11), localizada na parte direita da aplicação, conforme os mesmos vão sendo feitos. Nesta coluna também é possível visualizar o subtotal e o total da conta. O subtotal representa a soma dos valores de todos os pedidos feitos. O total representa o valor subtotal acrescido dos dez por cento do garçom. O sistema permite que o cliente opte pelo não pagamento desses dez por cento. Neste caso os valores total e subtotal da conta serão idênticos.

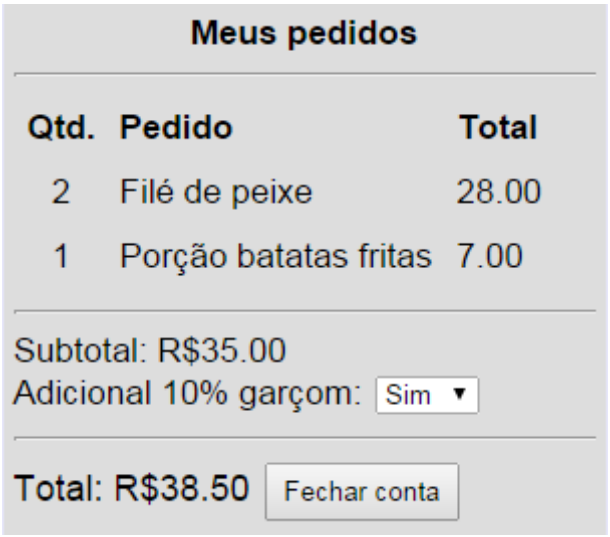

Figura 3.11 – Coluna de pedidos

(Fonte: Autor)

Para fechar a conta, o cliente deve tocar em “Fechar conta”. Esta opção só estará disponível após o usuário realizar ao menos um pedido. No momento em que o usuário opta por fechar a conta, o sistema abre um formulário de avaliação do atendimento (figura 3.12). São cinco as opções de avaliação: péssimo, fraco, regular, bom e ótimo. O usuário também

pode, opcionalmente, adicionar algum comentário que deseje. Todas as avaliações enviadas estarão visíveis para o usuário com perfil *admin*. Cada avaliação poderá então ser analisada e medidas podem ser tomadas de acordo com o resultado dessa análise, a critério do estabelecimento. O restaurante poderia, por exemplo, premiar os funcionários no caso de boas avaliações.

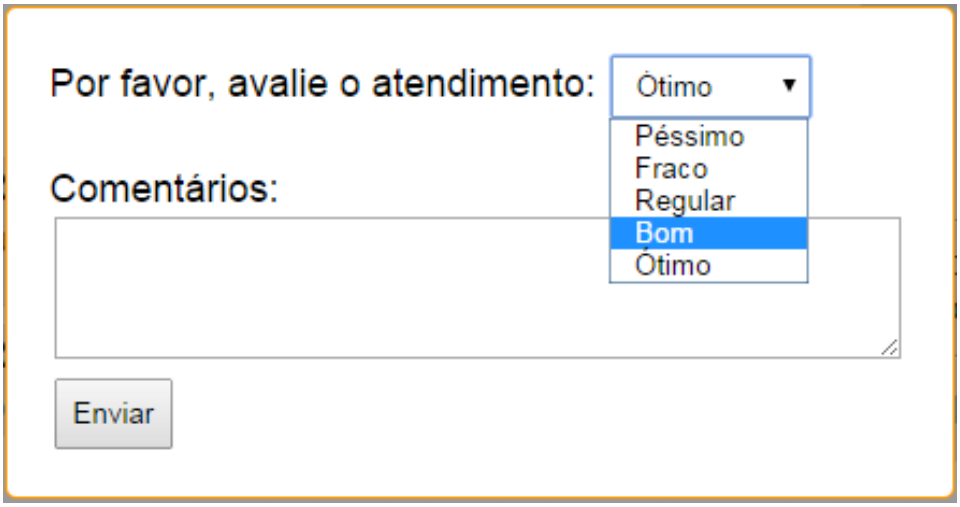

Figura 3.12 – Formulário de avaliação do atendimento

(Fonte: Autor)

A página de dúvidas (figura 3.13) é utilizada quando o usuário quer tirar alguma dúvida. Um exemplo seria o caso em que o cliente deseja solicitar o prato filé de peixe, que vem acompanhado de brócolis. Porém o cliente deseja que este prato venha com outro acompanhamento ao invés de brócolis. Neste caso o cliente poderia enviar a seguinte dúvida: “Seria possível substituir o brócolis por repolho?”. A dúvida é então inserida na tabela duvidas do banco de dados e será respondida pelo usuário com perfil *cozinha*. Logo abaixo na mesma página o usuário visualiza as respostas das dúvidas que enviou. Para o usuário não precisar ficar recarregando a página com o intuito de saber se a sua dúvida foi respondida ou não, foi criada uma função *JavaScript* que, via AJAX, faz constantemente uma solicitação de pesquisa no banco de dados ao PHP, que retorna todas as dúvidas daquele usuário que já foram respondidas.

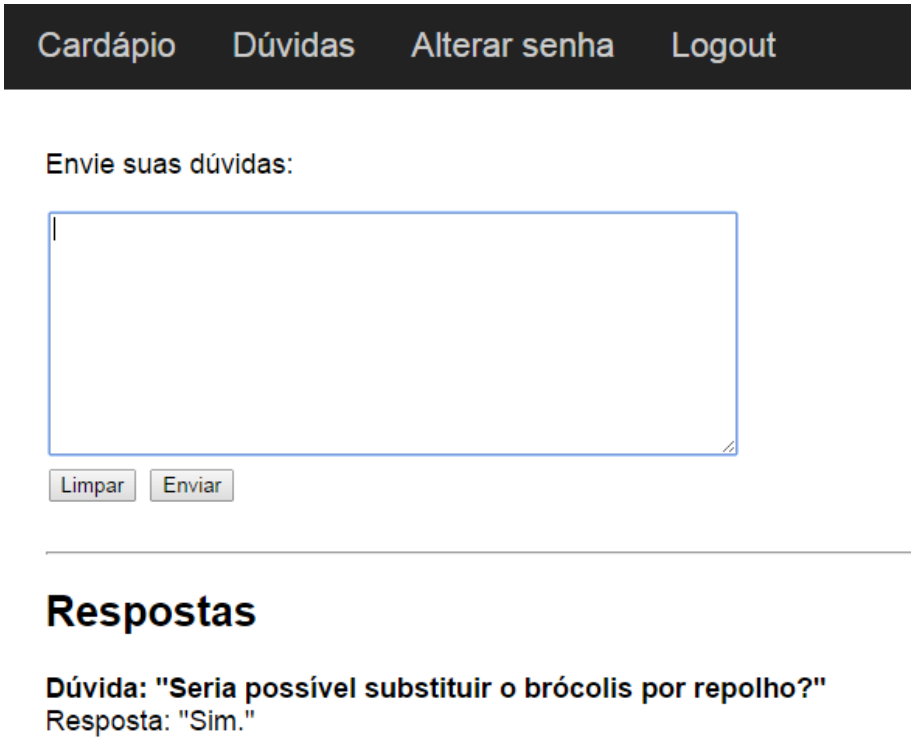

Figura 3.13 – Página de dúvidas

(Fonte: Autor)

A página de alteração de senha (figura 3.14) é utilizada pelo usuário quando o mesmo deseja alterar a sua senha. Essa página será utilizada, na maioria das vezes, quando o usuário esquecer a sua senha e solicitar uma nova senha ao sistema, visto que esta nova senha (gerada automaticamente) é aleatória e de difícil memorização. Para alterar a senha, o usuário deve informar a sua senha atual, a nova senha (que substituirá a senha atual) e uma confirmação da nova senha.

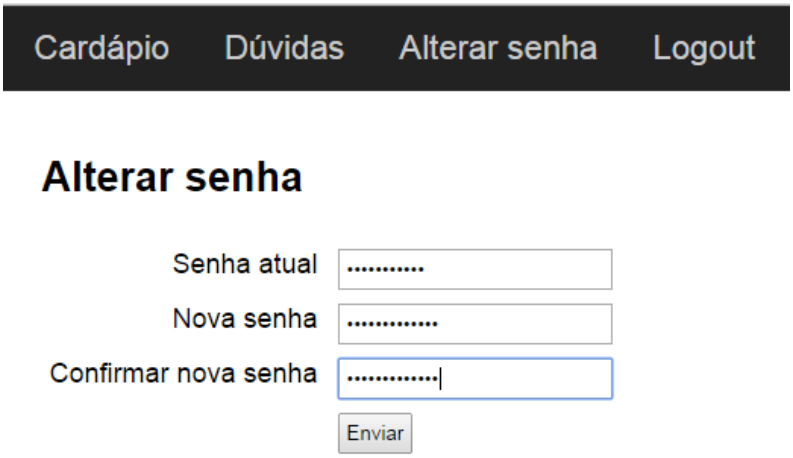

Figura 3.14 – Página de alteração de senha

(Fonte: Autor)

3.2.6 – Interface que implementa o perfil *cozinha*

O *login* de acesso ao sistema do usuário com perfil *cozinha* pode ser qualquer conjunto de caracteres (definido no momento de criação deste usuário pelo usuário de perfil *admin*), não sendo necessariamente um endereço de e-mail. Usuários com perfil *cozinha* só podem ser criados e alterados pelo usuário com perfil *admin*. A função principal desse usuário é atender pedidos e esclarecer dúvidas dos clientes. Esse usuário possui acesso a três páginas: lista de pedidos, lista de dúvidas e cardápio. A figura 3.15 mostra a página de lista de pedidos, que se trata da página inicial deste usuário.

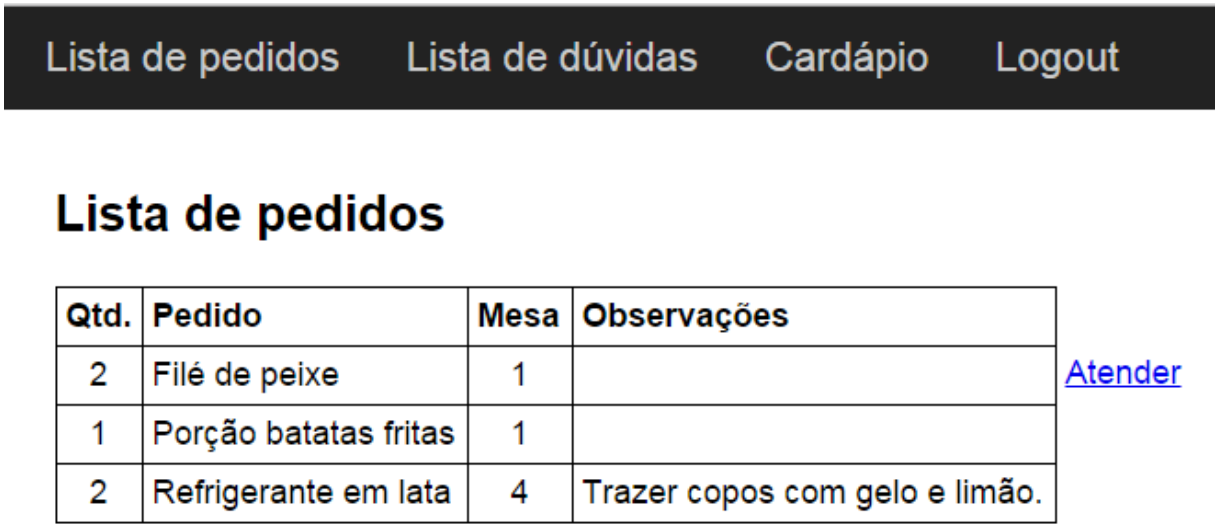

Figura 3.15 – Página de lista de pedidos

(Fonte: Autor)

A página de lista de pedidos é a principal página do usuário com perfil *cozinha*. Nesta página o usuário visualiza todos os pedidos feitos pelos clientes. Antes de carregar esta página o modelo faz um busca no banco de dados, trazendo o item pedido, quantidade solicitada, a mesa que solicitou o pedido e as observações do pedido de todos os pedidos que ainda não foram atendidos, por ordem de pedido. Para garantir que os pedidos mais antigos terão prioridade no atendimento, a aplicação só permite ao usuário de perfil *cozinha* atender ao pedido mais antigo. Para atender ao pedido basta clicar no *link* “Atender”. A aplicação irá então atualizar a situação do pedido para atendido e o respectivo pedido não será mais trazido na lista de pedidos. Após atender um pedido, o sistema libera o *link* para que o pedido seguinte na ordem de antiguidade seja atendido e assim por diante, até que todos os pedidos

tenham sido atendidos. O sistema checa constantemente se algum novo pedido foi enviado e atualiza automaticamente a lista de pedidos.

A página de lista de dúvidas (figura 3.16) corresponde à página na qual o usuário responde às dúvidas dos clientes. De maneira análoga ao que ocorre na lista de pedidos, esta página ordena as dúvidas da mais antiga para a mais recente, sendo que a aplicação impede que uma dúvida mais recente do que outra seja respondida antes. Para responder a uma dúvida basta clicar no *link* “Responder”. A aplicação irá então redirecionar o usuário para outra página, onde poderá escrever a resposta e enviá-la ao cliente. Caso uma dúvida já tenha sido respondida ou não precise mais ser esclarecida também é possível apaga-la clicando no *link* “Deletar”.

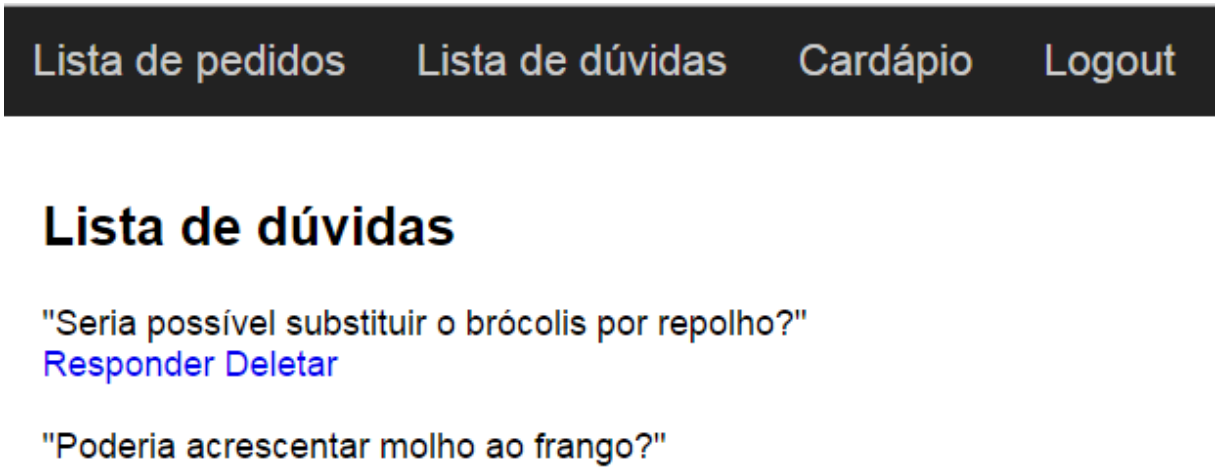

Figura 3.16 – Página de lista de dúvidas

(Fonte: Autor)

A última página é a página cardápio (figura 3.17). A página cardápio do usuário com perfil *cozinha* não pode ser confundida com a página de cardápio do usuário com perfil *cliente*. Enquanto esta mostra todos os itens do cardápio que o cliente pode pedir, aquela é usada para adicionar um item novo ao cardápio. Para adicionar um item é preciso informar a categoria do item a ser adicionado, nome do item, preço e descrição (sendo esta última informação opcional).

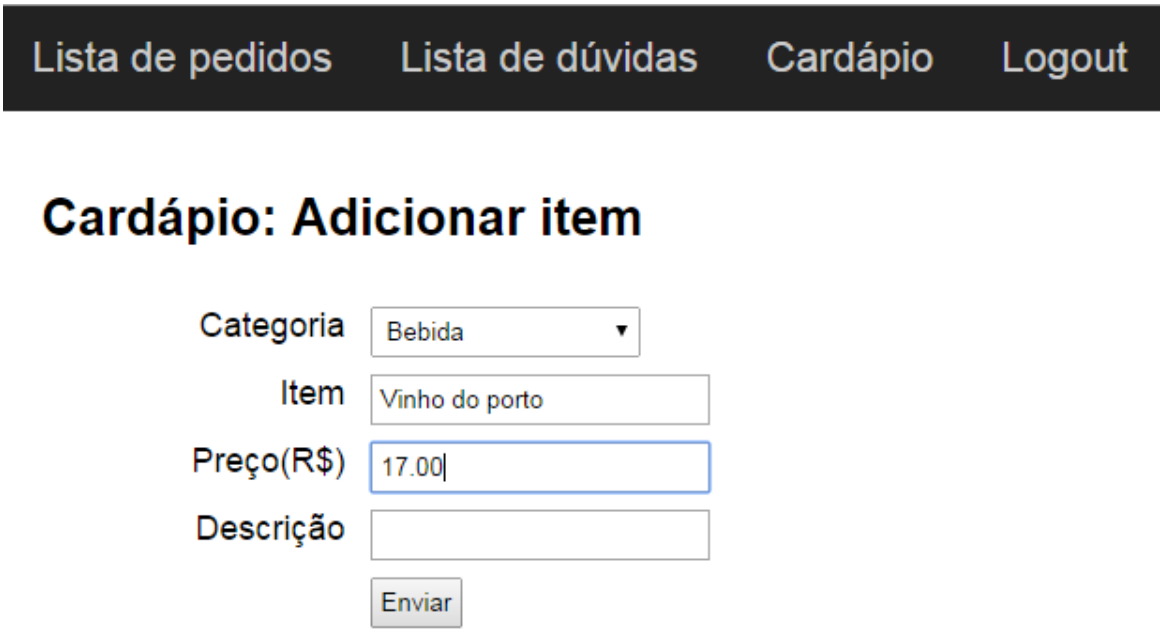

Figura 3.17 – Página cardápio

(Fonte: Autor)

3.2.7 – Interface que implementa o perfil *caixa*

Assim como o usuário com perfil *cozinha*, o usuário com perfil *caixa* só pode ser criado e alterado pelo usuário com perfil *admin*. Seu *login* de acesso ao sistema também não necessita ser um endereço de e-mail, podendo ser qualquer conjunto de caracteres. A única e principal função desse usuário é realizar o fechamento de contas dos clientes. O caixa possui acesso apenas à página de contas para fechamento, representada na figura 3.18.

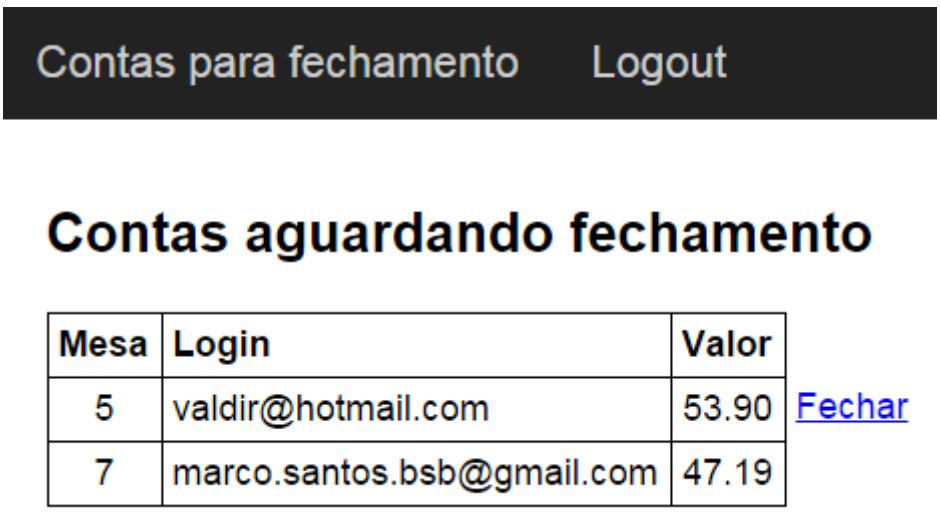

Figura 3.18 – Página de contas para fechamento

(Fonte: Autor)

A página contas para fechamento lista todas as contas de clientes que solicitaram o fechamento da conta. Esta página segue a mesma lógica da página de lista de pedidos do usuário com perfil *cozinha*. Antes de a página ser carregada o modelo faz um busca no banco de dados, trazendo a mesa que solicitou o fechamento da conta, o *login* do usuário que solicitou o fechamento da conta e o valor total da conta de todas as contas que ainda não foram fechadas, por ordem de solicitação. Para garantir que as contas mais antigas terão prioridade no fechamento, a aplicação só permite ao usuário fechar a conta mais antiga. Para fechar a conta basta clicar no *link* "Fechar". A aplicação irá então atualizar a situação da conta para fechada e a respectiva conta não será mais trazida na lista de contas. Após fechar uma conta, o sistema libera o *link* para que a conta seguinte na ordem de antiguidade seja atendida e assim por diante, até que todas as contas tenham sido fechadas. O sistema checa constantemente se alguma nova conta foi enviada e atualiza automaticamente as contas para fechamento.

3.2.8 – Interface que implementa o perfil *admin*

O último usuário do sistema é o usuário com perfil *admin*. Por motivos de segurança, este usuário é o único que não pode ser criado e nem alterado via sistema. Para que isso seja feito é necessário que os comandos de inserção (ou alteração, dependendo do caso) sejam executados diretamente no banco de dados. Este usuário, em geral, será o dono ou gerente do restaurante. Recomenda-se que não haja muitos usuários com este tipo de perfil, devendo ser criados no máximo dois. A principal função desse usuário é gerenciar as contas dos usuários com perfis *caixa* e *cozinha*. Esse usuário possui acesso a duas páginas: *admin*istrar usuários e avaliações. A figura 3.19 mostra a página de *admin*istração de usuários, que trata da página inicial deste usuário.

Na página de *admin*istração de usuários o usuário com perfil *admin* possui um formulário disponível, no qual ele pode adicionar algum novo usuário de perfil *caixa* ou *cozinha*. Neste formulário é esperado que o *admin* digite o *login* e a senha do novo usuário e selecione o perfil (*caixa* ou *cozinha*). Após isso basta clicar em "Enviar" para que o sistema faça o registro do usuário no banco de dados. Abaixo desse formulário, na mesma página, existe uma tabela que lista todos os usuários cadastrados com perfil *caixa* e *cozinha*. Esta tabela mostra o *login* do usuário e de qual perfil este faz parte. Para cada usuário listado são disponibilizados dois links, sendo um para deletar o usuário e outro para editá-lo. Caso o *link* "Editar" seja clicado, o usuário de perfil *admin* é redirecionado para um formulário onde pode

alterar o *login*, senha e/ou o perfil daquele usuário.

Figura 3.19 – Página de *admin*istração de usuários

(Fonte: Autor)

A página de lista de avaliações (figura 3.20) retorna todas as avaliações e comentários feitos pelos clientes acerca do atendimento. Importante ressaltar que esta lista é ordenada de maneira que as avaliações mais recentes estejam no topo e as mais antigas por último.

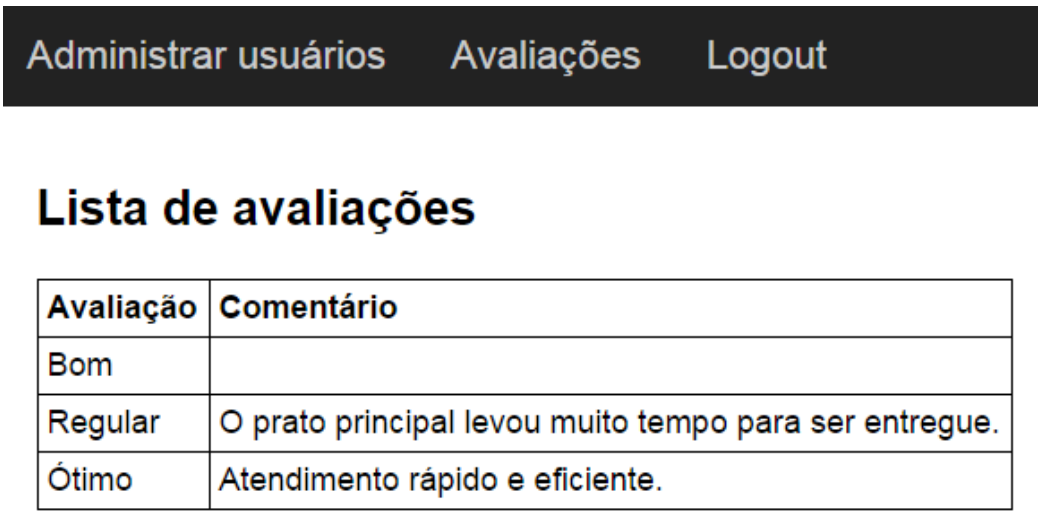

Figura 3.20 – Página de lista de avaliações

(Fonte: Autor)

Toda a implementação das interfaces que implementam os perfis acima expostos, assim como a comunicação com o hardware, foi realizada com o desenvolvimento da programação apresentando nos apêndices A, B e C.

CAPÍTULO 4 – APLICAÇÃO PRÁTICA DO MODELO DE SOLUÇÃO

Este capítulo tem como objetivo principal mostrar a viabilidade da proposta sugerida no trabalho. Será feita uma avaliação acerca dos resultados obtidos, quais os estabelecimentos que podem aplicar melhor a proposta, dificuldades encontradas na implementação e custos do modelo.

4.1 – Resultados

Fez-se uma estimativa de quanto tempo se pode economizar com a implementação deste projeto. Para tanto, foi considerado um espaço de quinze metros entre o balcão (onde o pedido do cliente é informado pelo garçom para preparo) e a mesa do cliente. Este espaço foi medido por meio de uma trena. A figura 4.1 apresenta um comparativo entre o tempo gasto no envio do pedido por meio do garçom e por meio do sistema. O tempo médio (em segundos) foi obtido utilizando-se um cronômetro.

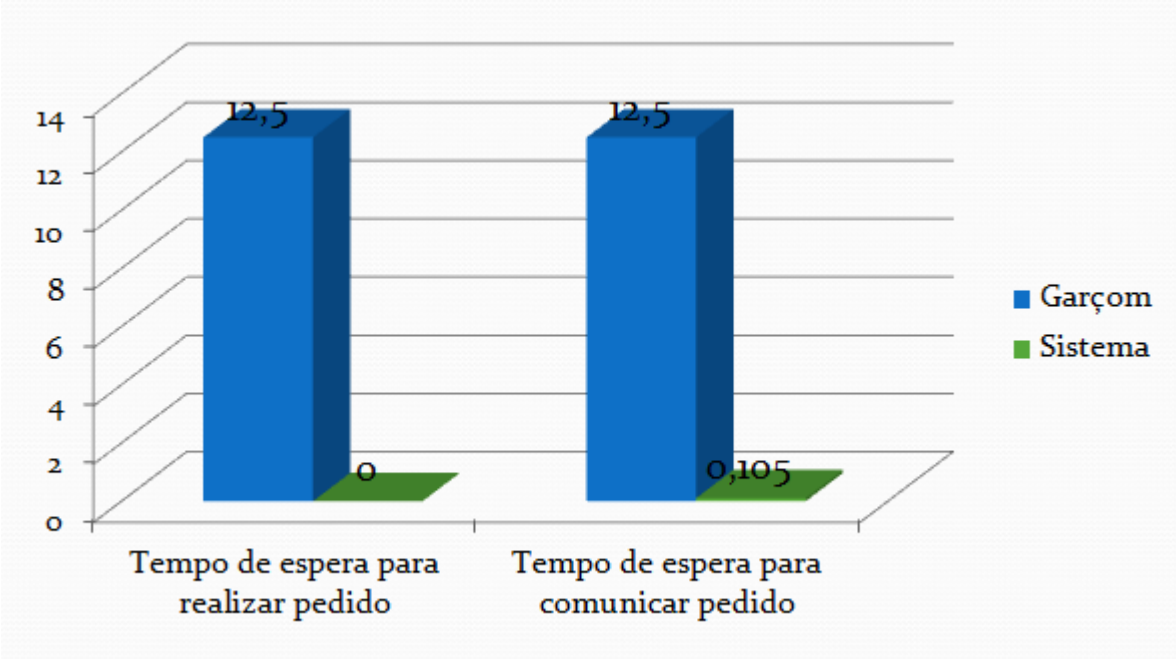

Figura 4.1 – Tempo

(Fonte: Autor)

Em um primeiro momento temos o tempo gasto para que o cliente possa realizar o pedido. O tempo decorrido entre o momento que o cliente chama o garçom e este chega à

mesa é de 12,5 segundos. Porém, fazendo uso do sistema, o cliente já pode pedir, sem ter que aguardar o garçom. Em um segundo momento temos o tempo gasto para que o pedido do cliente seja comunicado à cozinha. O garçom demora outros 12,5 segundos para retornar ao balcão e comunicar o pedido. Por outro lado o retardo em uma rede local é da ordem de milissegundos, isto é, o tempo gasto entre o momento no qual o cliente clica em "Adicionar" e o momento no qual o pedido é armazenado no banco de dados e o *buzzer* é acionado (sinalizando a chegada do pedido) é de 105 milissegundos.

Conclui-se, então, que o tempo economizado seria de aproximadamente 24,805 segundos. Não estão sendo levados outros fatores em consideração, como por exemplo, a possibilidade do garçom atender duas ou mais mesas antes de comunicar os pedidos no balcão. Neste caso, o tempo gasto no atendimento das demais mesas deveria ser somado, o que acarretaria um aumento significativo no tempo total.

4.2 - Apresentação da Área de Aplicação da Solução

Este projeto não apresenta muitas restrições quanto ao tipo de restaurante que pode implementá-lo, podendo ser utilizado pelos mais variados estabelecimentos, independentemente do perfil do *cliente*, tipos de pratos servidos, etc. Contudo, conforme será visto mais adiante neste capítulo, tem-se um custo razoável para a implementação deste projeto, principalmente por considerar que o restaurante é responsável por fornecer o *tablet* ao cliente. Assim sendo, sua viabilidade dependerá da capacidade do restaurante para atender com eficiência e rapidez os pedidos dos clientes e da disponibilidade de recursos para o investimento em tecnologia. Normalmente, quanto maior a quantidade de clientes que o restaurante recebe, menor é esta capacidade, sendo necessário contratar mais funcionários para conseguir atender os clientes apropriadamente. É geralmente para esse tipo de situação (quantidade demasiada de clientes) que este projeto terá maior utilidade. Para restaurantes de pequeno porte, com quantidade menor de clientes, o modelo proposto pode não ser necessário.

4.3 – Dificuldades Encontradas

Durante o desenvolvimento algumas partes da aplicação eram exibidas corretamente no navegador do *desktop*, porém eram incorretamente exibidas no navegador do *tablet.* Isso ocorreu principalmente com formulários, onde alguns campos eram posicionados em locais

onde não deveriam aparecer. Para solucionar este problema foi necessário rever e mudar algumas partes do código fonte, o que acabou acarretando retrabalho.

Outra dificuldade foi em fazer com que o sistema atualizasse a lista de pedidos da página do usuário com perfil *cozinha* sempre que o cliente enviasse um novo pedido. Esta página deveria ser desenvolvida de maneira que ao ser enviado um novo pedido o sistema adicionasse este pedido ao final da lista, sem a necessidade do usuário recarregar a página e sem a necessidade da página inteira precisar ser recarregada. A solução encontrada foi desenvolver uma função *JavaScript* para fazer uma busca dos pedidos via AJAX no banco de dados de tempos em tempos, comparar com os pedidos que já estão sendo exibidos e, caso fosse encontrado um pedido que ainda não está listado, adicioná-lo ao final da lista. Esta solução foi replicada em outras partes do sistema, como na página de lista de contas do usuário com perfil *caixa*.

4.4 – Custo dos componentes do projeto

A tabela 4.1 apresenta o custo estimado de cada dispositivo utilizado para implementar este projeto.

Tabela 4.1 – Custo estimado dos dispositivos

Quantidade	Nome	Preço (R$)
1	*Buzzer*	1,28
1	LED	0,12
1	Arduino	54,90
1	*Tablet*	1300,00
1	Notebook	1500,00
1	*Desktop*	1500,00
	Total:	**4356,30**

FONTE: Autor

O valor total presente na tabela representa o custo por restaurante, considerando o restaurante com apenas um *tablet* disponível. Ressalta-se que esse valor pode variar, dependendo da quantidade de *tablets* a ser adquirida pelo restaurante. O ideal é que haja pelo menos um *tablet* para cada mesa do estabelecimento, de maneira que nenhum cliente fique sem *tablet* no caso de todas as mesas estarem ocupadas. Os demais dispositivos não variam em quantidade.

O próximo e último capítulo discorrerá acerca das conclusões do trabalho, analisando se os objetivos foram alcançados, se os resultados são satisfatórios, vantagens e desvantagens do projeto, entre outras observações. Por fim serão dadas sugestões para trabalhos futuros, visando aprimorar o sistema já desenvolvido.

CAPÍTULO 5 – CONSIDERAÇÕES FINAIS

5.1 – Conclusões

Este projeto teve por objetivo principal desenvolver um sistema através do qual os clientes de restaurantes pudessem, por meio de um *tablet*, fazer seus pedidos sem o auxílio de um garçom. Nota-se que os objetivos inicialmente propostos foram alcançados, sendo que a aplicação dispõe para o cliente uma interface por meio da qual ele tem acesso a tudo de que necessita para fazer seu pedido. Além disso, a aplicação também permite ao usuário enviar dúvidas, acompanhar o valor da conta, solicitar o fechamento da conta, e, como apresentado, realizar os pedidos com rapidez, se comparado ao pedido feito através do garçom, entre outros.

As principais vantagens desta proposta de solução são proporcionar maior comodidade aos clientes, que não precisam se apressar no momento de escolher o pedido, e agilizar o pedido, visto que os pedidos são enviados direto para a cozinha, onde soa um alerta para auxiliar os responsáveis pelo atendimento a identificarem a chegada do mesmo. A parte de hardware responsável por emitir este alerta (composta basicamente por Arduino, LED e *buzzer*) é de fácil implementação e de baixo custo, como ficou demonstrado na implementação e desenvolvimento. Ademais, o Arduino possibilita que outras funcionalidades extras, tal como a exibição de mensagens em uma tela LCD, sejam adicionadas a partir da base deste modelo.

A principal desvantagem desta solução é a de possuir um custo global considerável, principalmente em função da quantidade de *tablets* a ser adquirida. Desta maneira a sua implementação é dificultada em restaurantes que não possuem problemas para conseguir atender adequadamente os clientes. Esta desvantagem poderá deixar de existir futuramente, já que o custo de componentes eletrônicos tende a cair com o tempo, viabilizando a implementação deste projeto em mais restaurantes. Além disso, os clientes poderiam também utilizar seus próprios *tablets* e *smartphones.*

A automação de processos vem se tornando uma tendência nos mais variados tipos de empreendimentos. Os restaurantes não fogem a essa regra, sendo que alguns dos principais deles já utilizam algum tipo de automação. Isso só tem a trazer vantagens, tanto para os donos dos estabelecimentos quanto para os próprios clientes.

5.2 – Sugestões para Trabalhos Futuros

Diversas funcionalidades ainda podem ser adicionadas a este projeto. Entre elas podemos destacar:

- Desenvolvimento de uma interface para *smartphones*: a aplicação poderia também possuir uma versão para *smartphones*, de maneira que o cliente pudesse acessá-la pelo seu próprio celular. A principal vantagem disso seria a diminuição dos custos do restaurante com a compra de *tablets* para os clientes.
- Cálculo e armazenamento do tempo gasto no atendimento de um pedido (incluindo-se aí o tempo gasto desde a solicitação, preparo na cozinha e entrega ao cliente): o sistema poderia calcular o tempo que foi gasto entre o envio de um pedido e o atendimento do mesmo e armazenar este dado. Desta forma poderia ser feita uma análise posterior acerca do tempo médio dos atendimentos e se o mesmo está sendo satisfatório ou não.
- Automatização do pagamento: poderia ser adicionada uma funcionalidade no sistema que permitisse ao próprio cliente pagar a conta por meio do número do cartão de crédito, de maneira similar ao pagamento de uma compra na *internet*. Desta maneira não haveria necessidade do cliente chamar o garçom com o intuito de efetuar o pagamento da conta (excetuando-se os casos em que o cliente deseja pagar em dinheiro).
- Possibilidade de o cliente fazer o pedido antecipadamente: poderia ser dada ao cliente a possibilidade de este comunicar o pedido antes de chegar ao restaurante, informando o horário que irá chegar. Desta forma, o cliente já teria o seu pedido pronto e uma mesa reservada ao chegar ao estabelecimento.
- Imagens dos itens do cardápio: o sistema poderia mostrar ao cliente, quando este tocasse em um item do cardápio, uma imagem do item selecionado. A imagem seria carregada juntamente com o formulário no qual o usuário define a quantidade desejada daquele item e envia o pedido.
- Por fim, o sistema pode ser aprimorado para mostrar simultaneidade entre os alarmes com avisos sonoros de pedidos em sequencia apresentados em tela display para acompanhamento tanto pelo usuário *cozinha* como pelo cliente.

REFERÊNCIAS BIBLIOGRÁFICAS

BRAY, Tim. **Extensible markup language (xml) 1.0.** 2004.

DEACON, John. **Model-view-controller (mvc) architecture.** 2009.

FLANAGAN, David. **JavaScript: The definitive guide: Activate your web pages.** " O'Reilly Media, Inc.", 2011.

FREEMAN, Elisabeth. **Use a cabeça!: HTML com CSS e HTML.** Alta books, 2008.

GARRETT, Jesse James. **Ajax: A new approach to web applications**. 2005.

MCROBERTS, Michael. **Arduino básico.** Novatec Editora, 2011.

PRESSMAN, Roger. **Engenharia de software.** McGraw Hill Brasil, 2011.

ROB, Peter; CORONEL, E. Carlos. **Sistemas de Banco de Dados-Projeto, Implementação e *Admin*istração**. São Paulo: Cengage Learning, 2011.

SILVA, Maurício. **jQuery A Biblioteca do Programador JavaScript.** São Paulo: Novatec Editora, 2008.

TANENBAUM, Andrew. **Redes de computadores.** Pearson Educación, 2012.

TANENBAUM, Andrew. **Sistemas distribuídos.** Pearson Educación, 2012.

WELLING, Luke; THOMSON, Laura. **PHP and MySQL Web development**. Sams Publishing, 2003.

Sites consultados:

Create your own MVC. Disponível em: <https://jream.com/lab> Acessado em: 25 de Julho de

2014

A teoria de motivação de Maslow. Disponível em: <http://ogerente.com/stakeholder/2007/04/03/a-teoria-de-motivacao-de-maslow/> Acessado em: 15 de Setembro de 2014

Tecnologia do restaurante Mangai. Disponível em: http://www.mangai.com.br/site/qualidade/tecnologia/ Acessado em: 16 de Setembro de 2014

Como funcionam os microcontroladores. Disponível em: <http://tecnologia.hsw.uol.com.br/microcontroladores.htm> Acessado em: 19 de Setembro de 2014

O que é o AJAX e como ele funciona. Disponível em: <http://codigofonte.uol.com.br/artigos/o-que-e-o-ajax-e-como-ele-funciona> Acessado em: 21 de Setembro de 2014

O que é XML? Disponível em: <http://www.tecmundo.com.br/programacao/1762-o-que-e-xml-.htm> Acessado em: 21 de Setembro de 2014

Servidor Web Apache. Disponível em: <http://softwarelivre.org/php/servidor-web-apache> Acessado em: 22 de Setembro de 2014

APÊNDICE

Apêndice A – Código do Microcontrolador Arduino

```
//A lista de código a seguir implementa a programação do Arduino, responsável por controlar
//o hardware do projeto.

#include <SPI.h>
#include <Ethernet.h>

//armazena o ultimo momento no qual o buzzer foi ativado
unsigned long previousMillis = 0;

int buzzerPin = 3;
int ledPin = 7;
int buttonPin = 6;
//variável utilizada para definer o intervalo de funcionamento do buzzer (no caso, 3 seg)
const long intervalo = 3000;
boolean buz = 0;

void setup() {

  Serial.begin(9600);
  //configura os pinos usados como entrada ou saída
  pinMode(buzzerPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
  pinMode(buttonPin, INPUT);

}

void loop() {
  unsigned long currentMillis = millis();

  if (digitalRead(buttonPin) == HIGH){
```

```
    digitalWrite(ledPin, LOW);
  }

  if (buz == 1){
    if(currentMillis - previousMillis >= intervalo) {
      digitalWrite(buzzerPin, LOW);
      buz = 0;
    }
  }

  //se recebeu dados vindos da porta serial, ligar o LED e ativar o buzzer.
  if (Serial.available() > 0) {
      digitalWrite(buzzerPin, HIGH);
      digitalWrite(ledPin, HIGH);
      Serial.println(Serial.read());
      buz = 1;
      previousMillis = currentMillis;
  }

}
```

Apêndice B – Comandos de Criação do Banco de Dados

//Os comandos a seguir implementam criação das tabelas do banco de dados conforme //modelo entidade relacionamento (MER) apresentado na seção 3.2.2.1.

```
CREATE DATABASE projeto_restaurante;


CREATE TABLE usuarios(
	user_id INT(11) NOT NULL AUTO_INCREMENT,
	login VARCHAR(32) NOT NULL,
	password CHAR(40) NOT NULL,
	perfil ENUM('admin','cozinha','caixa','cliente') NOT NULL DEFAULT 'cliente',
	mesa INT(2),
	PRIMARY KEY(user_id)
);

CREATE TABLE duvidas(
	duvida_id INT(11) NOT NULL AUTO_INCREMENT,
	fk_user_id INT(11) NOT NULL,
	duvida VARCHAR(500) NOT NULL,
	resposta VARCHAR(500) ,
	FOREIGN KEY (fk_user_id) REFERENCES usuarios (user_id) ON DELETE
CASCADE,
	PRIMARY KEY(duvida_id)
);


CREATE TABLE cardapio(
	item_id INT(11) NOT NULL AUTO_INCREMENT,
	item VARCHAR(25) NOT NULL,
	preco DECIMAL(6,2) NOT NULL,
	descricao varchar(60),
	categoria ENUM('bebida','prato','acompanhamento','sobremesa') NOT NULL,
	PRIMARY KEY(item_id)
);

CREATE TABLE pedidos(
	pedido_id INT(11) NOT NULL AUTO_INCREMENT,
	fk_user_id INT(11) NOT NULL,
	fk_item_id INT(11) NOT NULL,
	observacao varchar(70),
	quantidade INT(1),
	valor_total DECIMAL(6,2) NOT NULL,
	atendido ENUM('S', 'N') NOT NULL DEFAULT 'N',
	FOREIGN KEY (fk_user_id) REFERENCES usuarios (user_id) ON DELETE
CASCADE,
	FOREIGN KEY (fk_item_id) REFERENCES cardapio (item_id) ON DELETE
```

```
CASCADE,
        PRIMARY KEY(pedido_id)
);

CREATE TABLE contas (
   conta_id INT(11) NOT NULL AUTO_INCREMENT,
   fk_user_id INT(11) NOT NULL,
   valor DECIMAL(6,2) NOT NULL,
   fechada ENUM('S','N') NOT NULL DEFAULT 'N',
   FOREIGN KEY (fk_user_id) REFERENCES usuarios (user_id) ON DELETE CASCADE,
   PRIMARY KEY(conta_id)
);

CREATE TABLE avaliacoes (
   avaliacao_id INT(11) NOT NULL AUTO_INCREMENT,
   avaliacao ENUM('Péssimo','Fraco','Regular','Bom','Ótimo') NOT NULL,
   comentario VARCHAR(400),
   PRIMARY KEY(avaliacao_id)
);
```

Apêndice C – Código do Sistema

//A listagem de código abaixo implementa todo o sistema com seus módulos, relacionando os //diferentes perfis desenvolvidos conforme apresentados neste trabalho.

Pasta “config”:

- database.php:

```
<?php

	define('DB_TYPE', 'mysql');
	define('DB_HOST', 'localhost');
	define('DB_NAME', 'projeto_restaurante');
	define('DB_USER', 'root');
	define('DB_PASSWORD', '');
```

- paths.php:

```
<?php

	define('URL', '/restaurante/');
```

Pasta “controllers”:

- alteraSenha.php:

```
<?php

class alteraSenha extends Controller {

	public function __construct() {
		parent::__construct();

		Session::init();
		$logged = Session::get('loggedIn');
		$perfil = Session::get('perfil');

		if($logged == false || ($perfil != 'admin' && $perfil != 
'cliente')){
			Session::destroy();
			header('location:'.URL.'login');
			exit;
		}

		$this->view->js = array('alteraSenha/js/default.js');
		$this->view->css = 'alteraSenha/css/style.css';
	}
```

```
	public function index(){
		$this->view->render('alteraSenha/index');
	}

	public function alterar(){
		$this->model->alterar();
	}

}
```

- avaliações.php:

```
<?php

class Avaliacoes extends Controller {

	public function __construct(){
		parent::__construct();
		Session::init();
		$logged = Session::get('loggedIn');
		$perfil = Session::get('perfil');

		if($logged == false || $perfil != 'admin'){
			Session::destroy();
			header('location:'.URL.'login');
			exit;
		}
		$this->view->css = 'avaliacoes/css/style.css';
	}

	public function index(){
		$this->view->avaliacoesList = $this->model->avaliacoesList();
		$this->view->render('avaliacoes/index');
	}

}
```

- cardápio.php:

```
<?php

class cardapio extends Controller {

	public function __construct() {
		parent::__construct();
		Session::init();
		$logged = Session::get('loggedIn');
		$perfil = Session::get('perfil');

		if($logged == false || ($perfil != 'admin' && $perfil !=
'cozinha')){
			Session::destroy();
			header('location:'.URL.'login');
			exit;
		}
```

```
            $this->view->css = 'cardapio/css/style.css';
      }

      public function index(){
            $this->view->render('cardapio/index');
      }

      public function adicionar(){
            $this->model->adicionar();
      }

}
```

- duvidas.php:

```
<?php

class Duvidas extends Controller {

      public function __construct() {
            parent::__construct();
            Session::init();
            $logged = Session::get('loggedIn');
            $perfil = Session::get('perfil');

            if($logged == false || ($perfil != 'admin' && $perfil !=
'cliente')){
                  Session::destroy();
                  header('location:'.URL.'login');
                  exit;
            }

            $this->view->js =
array('duvidas/js/default.js','duvidas/js/validaEnvio.js');
            $this->view->css = 'duvidas/css/style.css';
      }

      public function index(){
            $this->view->render('duvidas/index');
      }

      public function duvidaInsert(){
            $this->model->duvidaInsert();;
      }

      public function respostasList(){
            $this->model->respostasList();
      }

}
```

- erro.php:

```
<?php
```

```
class Erro extends Controller {

	function __construct() {
		parent::__construct();

	}

	function index(){
		$this->view->msg = 'Esta página não existe';
		$this->view->render('erro/index');
	}

}
```

- index.php:

```
<?php

class Index extends Controller {

	public function __construct(){
		parent::__construct();
		Session::init();
		$logged = Session::get('loggedIn');
		$perfil = Session::get('perfil');

		if($logged == false || ($perfil != 'admin' && $perfil != 
'cliente')){
			Session::destroy();
			header('location:'.URL.'login');
			exit;
		}

		$this->view->js = 
array('index/js/default.js','index/js/mostrarItem.js','index/js/mostrarForm
Avaliacao.js');
		$this->view->css = 'index/css/style.css';
	}

	public function index(){
		$this->view->render('index/index');
	}

	public function getCardapio(){
		$this->model->getCardapio();
	}

	public function getPedidos(){
		$this->model->getPedidos();
	}

	public function getStatusConta(){
		$this->model->getStatusConta();
	}

	public function getItem(){
		$this->model->getItem();
```

```
	}

	public function cadastrarPedido(){
		$this->model->cadastrarPedido();
	}

	public function fecharConta(){
		$this->model->fecharConta();
	}

}
```

- listaContas.php:

```
<?php

class listaContas extends Controller {

	public function __construct() {
		parent::__construct();
		Session::init();
		$logged = Session::get('loggedIn');
		$perfil = Session::get('perfil');

		if($logged == false || ($perfil != 'admin' && $perfil != 
'caixa')){
			Session::destroy();
			header('location:'.URL.'login');
			exit;
		}

		$this->view->js = array('listaContas/js/default.js');
		$this->view->css = 'listaContas/css/style.css';
	}

	public function index(){
		$this->view->render('listaContas/index');
	}

	public function contasList(){
		$this->model->contasList();
	}

	public function fecharPedido($conta_id){
		$this->model->fecharPedido($conta_id);
	}

}
```

- listaDuvidas.php:

```
<?php

class listaDuvidas extends Controller {
```

```
	public function __construct() {
		parent::__construct();
		Session::init();
		$logged = Session::get('loggedIn');
		$perfil = Session::get('perfil');

		if($logged == false || ($perfil != 'admin' && $perfil != 
'cozinha')){
			Session::destroy();
			header('location:'.URL.'login');
			exit;
		}

		$this->view->js = array('listaDuvidas/js/default.js');
		$this->view->css = 'listaDuvidas/css/style.css';
	}

	public function index(){
		$this->view->render('listaDuvidas/index');
	}

	public function duvidasList(){
		$this->model->duvidasList();
	}

	public function response($duvida_id){
		$this->view->duvida = $this->model-
>duvidaSingleList($duvida_id);
		$this->view->render('listaDuvidas/response');
	}

	public function responseSave($duvida_id){
		$data = array();
		$data['duvida_id'] = $duvida_id;
		$data['resposta'] = $_POST['resposta'];

		$this->model->responseSave($data);
		header('location:'.URL.'listaDuvidas');
	}

	public function delete($duvida_id){
		$this->model->delete($duvida_id);
	}

}
```

- listaPedidos.php:

```
<?php

class listaPedidos extends Controller {

	function __construct() {
		parent::__construct();
		Session::init();
		$logged = Session::get('loggedIn');
		$perfil = Session::get('perfil');
```

```
            if($logged == false || ($perfil != 'admin' && $perfil !=
'cozinha')){
                  Session::destroy();
                  header('location:'.URL.'login');
                  exit;
            }

            $this->view->js = array('listaPedidos/js/default.js');
            $this->view->css = 'listaPedidos/css/style.css';
      }

      function index(){
            $this->view->render('listaPedidos/index');
      }

      public function pedidosList(){
            $this->model->pedidosList();
      }

      public function atenderPedido($pedido_id){
            $this->model->atenderPedido($pedido_id);
      }

}
```

- login.php:

```
<?php

class Login extends Controller {

      public function __construct(){
            parent::__construct();
      }

      public function index(){
            $this->view->logar = '';
            $this->view->render('login/index',true);
      }

      public function run(){
            $this->view->logar = $this->model->run();
            $this->view->render('login/index',true);
      }

      public function cadastrar(){
            $this->model->cadastrar();
      }

      public function recuperarSenha(){
            $this->model->recuperarSenha();
      }

}
```

- logout.php:

```php
<?php

class Logout extends Controller {

	public function __construct(){
		parent::__construct();
		Session::init();
	}

	public function index(){
		$this->model->logout();
	}

}
```

- user.php:

```php
<?php

class User extends Controller {

	public function __construct(){
		parent::__construct();
		Session::init();
		$logged = Session::get('loggedIn');
		$perfil = Session::get('perfil');

		if($logged == false || $perfil != 'admin'){
			Session::destroy();
			header('location:'.URL.'login');
			exit;
		}

		$this->view->css = 'user/css/default.css';

	}

	public function index(){
		$this->view->userList = $this->model->userList();
		$this->view->render('user/index');
	}

	public function create(){
		$data = array();
		$data['login'] = $_POST['login'];
		$data['password'] = $_POST['password'];
		$data['perfil'] = $_POST['perfil'];

		$this->model->create($data);
		header('location:'.URL.'user');
	}

	public function edit($id){
		$this->view->user = $this->model->userSingleList($id);
		$this->view->render('user/edit');
	}
```

```
	public function editSave($id){
		$data = array();
		$data['user_id'] = $id;
		$data['login'] = $_POST['login'];
		$data['password'] = $_POST['password'];
		$data['perfil'] = $_POST['perfil'];

		$this->model->editSave($data);
		header('location:'.URL.'user');
	}

	public function delete($id){
		$this->model->delete($id);
		header('location:'.URL.'user');
	}

}
```

Pasta “libs”:

- bootstrap.php:

```
<?php

class Bootstrap {

	function __construct() {

		//recupera e divide a URL
		$url = isset($_GET['url']) ? $_GET['url'] : null;
		$url = rtrim($url, '/');
		$url = explode('/', $url);

		//caso nenhum controlador tenha sido informado, chama o
controlador padrão
		if (empty($url[0])) {
			require 'controllers/index.php';
			$controller = new Index();
			$controller->index();
			return false;
		}

		//chama o controlador
		$file = 'controllers/' . $url[0] . '.php';
		if (file_exists($file)) {
			require $file;
		} else {
			$this->erro();
		}

		@$controller = new $url[0];
		$controller->loadModel($url[0]);

		//chama os métodos
		if (isset($url[2])) {
			if (method_exists($controller, $url[1])) {
```

```
                        $controller->{$url[1]}($url[2]);
                  } else {
                        $this->erro();
                  }
            } else {
                  if (isset($url[1])) {
                        if (method_exists($controller, $url[1])) {
                              $controller->{$url[1]}();
                        } else {
                              $this->erro();
                        }
                  } else {
                        $controller->index();
                  }
            }


      }

      //chama a página de erro caso algum controlador ou método não tenha
sido encontrado
      function erro() {
            require 'controllers/erro.php';
            $controller = new Erro();
            $controller->index();
            return false;
      }

}
```

- controller.php:

```
<?php

class Controller {

      function __construct() {
            $this->view = new View();
      }

      public function loadModel($name){

            //chama o modelo
            $path = 'models/'.$name.'_model.php';

            if(file_exists($path)){
                  require 'models/'.$name.'_model.php';

                  $modelName = $name.'_model';
                  $this->model = new $modelName;
            }
      }

}
```

- database.php:

```
<?php

class Database extends PDO {

	public function __construct(){
		//estabelece uma conexão com o banco de dados

	parent::__construct(DB_TYPE.':host='.DB_HOST.';dbname='.DB_NAME,DB_US
ER,DB_PASSWORD);
	}

}
```

- model.php:

```
<?php

class Model {

	function __construct() {
		$this->db = new Database();
	}

}
```

- session.php:

```
<?php

class Session {

	public static function init(){
		@session_start();
	}

	public static function set($key,$value){
		$_SESSION[$key] = $value;
	}

	public static function get($key){
		if (isset($_SESSION[$key])){
			return $_SESSION[$key];
		}
	}

	public static function destroy(){
		session_destroy();
	}

}
```

- view.php:

```
<?php

class View {

	function __construct() {
		//echo 'Esta é a view <br>';
	}

	public function render($name, $noInclude = false){

		//constrói a visão no modelo cabeçalho, conteúdo e rodapé
		//caso o parâmetro $noInclude seja true, não traz o cabeçalho e
rodapé
		if($noInclude == true){
			require_once 'views/'.$name.'.php';
		}
		else{
			require_once 'views/header.php';
			require_once 'views/'.$name.'.php';
			require_once 'views/footer.php';
		}
	}
}
```

Pasta “models”:

- alteraSenha_model.php:

```
<?php

class alteraSenha_model extends Model {

	public function __construct(){
		parent::__construct();
	}

	public function alterar(){

		$user_id = Session::get('user_id');
		$current_password = $_POST['current-password'];
		$new_password = $_POST['new-password'];
		$confirm_password = $_POST['confirm-password'];

		$sth = $this->db->prepare("SELECT * FROM usuarios WHERE user_id
= :user_id AND password = SHA1(:password)");
		$sth->execute(array(
			':user_id' => $user_id,
			':password' => $current_password
		));

		$count = $sth->rowCount();

		if($count > 0){
```

```
                if($new_password == $confirm_password){
                        $sth = $this->db->prepare("UPDATE usuarios SET
password = SHA1(:password) WHERE user_id = :user_id");
                        $sth->execute(array(
                                ':user_id' => $user_id,
                                ':password' => $new_password
                        ));

                        echo '1';
                }
        }
        else{
                echo '0';
        }
    }

}
```

- avaliações_model.php:

```
<?php

class avaliacoes_model extends Model {

    public function __construct(){
        parent::__construct();
    }

    public function avaliacoesList(){
        $sth = $this->db->prepare('SELECT avaliacao,comentario FROM
avaliacoes ORDER BY avaliacao_id DESC');
        $sth->execute();
        return $sth->fetchAll();
    }

}
```

- cardápio_model.php:

```
<?php

class cardapio_model extends Model {

    public function __construct(){
        parent::__construct();
    }

    public function adicionar(){
        $categoria = $_POST['categoria'];
        $item = $_POST['item'];
        $preco = $_POST['preco'];
        $descricao = $_POST['descricao'];
```

```
            $sth = $this->db->prepare('INSERT INTO cardapio
(item,preco,descricao,categoria) VALUES
(:item,:preco,:descricao,:categoria)');
            $sth->execute(array(
                  ':item' => $item,
                  ':preco' => $preco,
                  ':descricao' => $descricao,
                  ':categoria' => $categoria
            ));

            header('location:'.URL.'cardapio');
      }

}
```

- duvidas_model.php:

```
<?php

class duvidas_model extends Model {

      public function __construct(){
            parent::__construct();
      }

      public function duvidaInsert() {
            $duvida = utf8_decode($_POST['duvida']);
            $user_id = $_POST['user_id'];

            $sth = $this->db->prepare('INSERT INTO duvidas
(fk_user_id,duvida) VALUES (:user_id,:duvida)');
            $sth->execute(array(
                  ':duvida' => $duvida,
                  ':user_id' => $user_id
            ));

            //header('location:'.URL.'duvidas');

      }

      public function respostasList() {
            $sth = $this->db->prepare('SELECT duvida_id,duvida,resposta
FROM duvidas WHERE fk_user_id = :fk_user_id AND resposta IS NOT NULL');
            $sth->execute(array(
                  ':fk_user_id' => Session::get('user_id')
            ));
            $valores = $sth->fetchAll();
            $respostas = array();
            $count = 0;
            foreach($valores as $key => $value){
                  $respostas[$count]['duvida_id'] =
utf8_encode($value['duvida_id']);
                  $respostas[$count]['duvida'] =
utf8_encode($value['duvida']);
                  $respostas[$count]['resposta'] =
utf8_encode($value['resposta']);
                  $count++;
            }
```

```
            echo json_encode($respostas);
      }

}
```

- index_model.php:

```
<?php

class index_model extends Model {

      public function __construct(){
            parent::__construct();
      }

      public function getCardapio() {
            $sth = $this->db->prepare('SELECT * FROM cardapio');
            $sth->execute();
            $valores = $sth->fetchAll();
            $itens = array();
            $count = 0;
            foreach($valores as $key => $value){
                  $itens[$count]['item_id'] = $value['item_id'];
                  $itens[$count]['item'] = utf8_encode($value['item']);
                  $itens[$count]['preco'] = $value['preco'];
                  $itens[$count]['descricao'] =
utf8_encode($value['descricao']);
                  $itens[$count]['categoria'] =
utf8_encode($value['categoria']);
                  $count++;
            }
            echo json_encode($itens);

      }

      public function getPedidos() {
            $sth = $this->db->prepare('SELECT pedidos.quantidade AS
quantidade,cardapio.item AS item,pedidos.valor_total AS valor_total FROM
pedidos INNER JOIN cardapio ON pedidos.fk_item_id = cardapio.item_id WHERE
fk_user_id = :fk_user_id');
            $sth->execute(array(
                  ':fk_user_id' => Session::get('user_id')
            ));
            $valores = $sth->fetchAll();
            $qtd = $sth->rowCount();

            if($qtd != 0){
                  $pedidos = array();
                  $count = 0;
                  foreach($valores as $key => $value){
                        $pedidos[$count]['quantidade'] =
$value['quantidade'];
                        $pedidos[$count]['item'] =
utf8_encode($value['item']);
                        $pedidos[$count]['valor_total'] =
$value['valor_total'];
                        $count++;
                  }
```

```
                echo json_encode($pedidos);
            }
            else {
                echo '0';
            }
        }

        public function getStatusConta() {
            $sth = $this->db->prepare('SELECT conta_id FROM contas WHERE
fk_user_id = :fk_user_id AND fechada = "N"');
            $sth->execute(array(
                ':fk_user_id' => Session::get('user_id')
            ));
            $data = $sth->fetch();
            $count = $sth->rowCount();
            if($count > 0){
                echo '1';
            }
            else{
                echo '0';
            }
        }

        public function getItem(){
            $item_id = $_POST['item_id'];

            $sth = $this->db->prepare('SELECT * FROM cardapio WHERE item_id
= :item_id');
            $sth->execute(array(
                ':item_id' => $item_id
            ));
            $valores = $sth->fetch();
            $valores_json = array();

            $valores_json['item_id'] = $valores['item_id'];
            $valores_json['item'] = utf8_encode($valores['item']);
            $valores_json['preco'] = $valores['preco'];
            $valores_json['descricao'] =
utf8_encode($valores['descricao']);
            $valores_json['categoria'] =
utf8_encode($valores['categoria']);

            echo json_encode($valores_json);
        }

        public function cadastrarPedido(){

            $sth = $this->db->prepare('SELECT conta_id FROM contas WHERE
fk_user_id = :fk_user_id AND fechada = "N"');
            $sth->execute(array(
                ':fk_user_id' => Session::get('user_id')
            ));

            $count = $sth->rowCount();

            if($count == 1){
                echo '1';
            }
            else{

                $item_id = $_POST['item-id'];
```

```
                $item = $_POST['item'];
                $item_valor = $_POST['item-valor'];
                $quantidade = $_POST['quantidade'];
                $observacao = utf8_decode($_POST['observacao']);
                $valor_total = $item_valor * $quantidade;

                if($observacao != 'Observações...'){
                      $sth = $this->db->prepare('INSERT INTO pedidos
(fk_user_id,fk_item_id,observacao,quantidade,valor_total) VALUES
(:fk_user_id,:fk_item_id,:observacao,:quantidade,:valor_total)');
                      $sth->execute(array(
                            ':fk_user_id' => Session::get('user_id'),
                            ':fk_item_id' => $item_id,
                            ':observacao' => $observacao,
                            ':quantidade' => $quantidade,
                            ':valor_total' => $valor_total
                      ));
                }
                else{
                      $sth = $this->db->prepare('INSERT INTO pedidos
(fk_user_id,fk_item_id,quantidade,valor_total) VALUES
(:fk_user_id,:fk_item_id,:quantidade,:valor_total)');
                      $sth->execute(array(
                            ':fk_user_id' => Session::get('user_id'),
                            ':fk_item_id' => $item_id,
                            ':quantidade' => $quantidade,
                            ':valor_total' => $valor_total
                      ));
                }

                $data = array('quantidade' => $quantidade, 'item' =>
$item, 'valor_total' => $valor_total);

                $port = fopen("COM4", "w");
                usleep(250000);
                fwrite($port,"ligar");
                fclose($port);

                echo json_encode($data);
          }
     }

     public function fecharConta(){
          $avaliacao = $_POST['avaliacao'];
          $comentario = $_POST['comentario'];
          $total = $_POST['total'];

          $sth = $this->db->prepare('INSERT INTO contas
(fk_user_id,valor) VALUES (:fk_user_id,:valor)');
          $sth->execute(array(
                ':fk_user_id' => Session::get('user_id'),
                ':valor' => $total
          ));

          $sth = $this->db->prepare('INSERT INTO avaliacoes
(avaliacao,comentario) VALUES (:avaliacao,:comentario)');
          $sth->execute(array(
                ':avaliacao' => $avaliacao,
                ':comentario' => $comentario
          ));
```

```
            header('location:'.URL.'index');
      }

}
```

- listaContas_model.php:

```
<?php

class listaContas_model extends Model {

      public function __construct(){
            parent::__construct();
      }

      public function contasList(){
            $sth = $this->db->prepare('SELECT usuarios.mesa AS
mesa,usuarios.login AS login,contas.conta_id AS conta_id,contas.valor AS
valor FROM usuarios INNER JOIN contas ON usuarios.user_id =
contas.fk_user_id WHERE fechada = "N"');
            $sth->execute();
            $valores = $sth->fetchAll();
            $contas = array();
            $count = 0;
            foreach($valores as $key => $value){
                  $contas[$count]['conta_id'] = $value['conta_id'];
                  $contas[$count]['mesa'] = $value['mesa'];
                  $contas[$count]['login'] = utf8_encode($value['login']);
                  $contas[$count]['valor'] = $value['valor'];
                  $count++;
            }
            echo json_encode($contas);
      }

      public function fecharPedido($conta_id){

            $sth = $this->db->prepare('SELECT fk_user_id FROM contas WHERE
conta_id = "'.$conta_id.'"');
            $sth->execute(array(':conta_id' => $conta_id));
            $result = $sth->fetch();

            $sth = $this->db->prepare('UPDATE contas SET fechada = "S"
WHERE conta_id = "'.$conta_id.'"');
            $sth->execute(array(':conta_id' => $conta_id));

            $sth = $this->db->prepare('DELETE FROM pedidos WHERE fk_user_id
= :fk_user_id');
            $sth->execute(array(
                  ':fk_user_id' => $result['fk_user_id']
            ));

            $sth = $this->db->prepare("UPDATE usuarios SET mesa = NULL
WHERE user_id = :user_id");
            $sth->execute(array(
                  ':user_id' => $result['fk_user_id']
            ));

            header('location:'.URL.'listaContas');
```

```
	}

}
```

- listaDuvidas_model.php:

```
<?php

class listaDuvidas_model extends Model {

	public function __construct(){
		parent::__construct();
	}

	public function duvidasList() {
		$sth = $this->db->prepare('SELECT duvida_id,duvida FROM duvidas
WHERE resposta IS NULL ORDER BY duvida_id');
		$sth->execute();
		$valores = $sth->fetchAll();
		$duvidas = array();
		$count = 0;
		foreach($valores as $key => $value){
			$duvidas[$count]['duvida_id'] = $value['duvida_id'];
			$duvidas[$count]['duvida'] =
utf8_encode($value['duvida']);
			$count++;
		}
		echo json_encode($duvidas);
	}

	public function duvidaSingleList($duvida_id){
		$sth = $this->db->prepare('SELECT duvida_id,duvida FROM duvidas
where duvida_id = :duvida_id');
		$sth->execute(array(':duvida_id' => $duvida_id));
		return $sth->fetch();
	}

	public function responseSave($data){

		$sth = $this->db->prepare('UPDATE duvidas SET
							resposta = :resposta
							WHERE duvida_id =
:duvida_id');
		$sth->execute(array(
					':duvida_id' => $data['duvida_id'],
					':resposta' => $data['resposta'],
					));
	}

	public function delete($duvida_id){
		$sth = $this->db->prepare('DELETE FROM duvidas WHERE duvida_id
= "'.$duvida_id.'"');
		$sth->execute(array(':duvida_id' => $duvida_id));
		header('location:'.URL.'listaDuvidas');
	}
```

```
}
```

- listaPedidos_model.php:

```
<?php

class listaPedidos_model extends Model {

	public function __construct(){
		parent::__construct();
	}

	public function pedidosList() {

		$sth = $this->db->prepare('SELECT p.pedido_id AS pedido_id,
c.item AS item, p.observacao AS observacao, p.quantidade AS quantidade,
u.mesa AS mesa FROM cardapio AS c INNER JOIN pedidos AS p ON c.item_id =
p.fk_item_id INNER JOIN usuarios AS u ON u.user_id = p.fk_user_id WHERE
atendido = "N" ORDER BY pedido_id');
		$sth->execute();
		$valores = $sth->fetchAll();
		$pedidos = array();
		$count = 0;
		foreach($valores as $key => $value){
			$pedidos[$count]['pedido_id'] = $value['pedido_id'];
			$pedidos[$count]['item'] = utf8_encode($value['item']);
			$pedidos[$count]['observacao'] =
utf8_encode($value['observacao']);
			$pedidos[$count]['quantidade'] = $value['quantidade'];
			$pedidos[$count]['mesa'] = $value['mesa'];
			$count++;
		}
		echo json_encode($pedidos);
	}

	public function atenderPedido($pedido_id){

		$sth = $this->db->prepare('UPDATE pedidos SET atendido = "S"
WHERE pedido_id = "'.$pedido_id.'"');
		$sth->execute(array(':pedido_id' => $pedido_id));
		header('location:'.URL.'listaPedidos');
	}

}
```

- login_model.php:

```
<?php

class login_model extends Model {
```

```
	public function __construct(){
		parent::__construct();
	}

	public function run(){

		$login = $_POST['login'];
		$password = $_POST['password'];
		$mesa = $_POST['mesa'];

		$sth = $this->db->prepare("SELECT user_id, perfil FROM usuarios
WHERE login = :login AND password = SHA1(:password)");
		$sth->execute(array(
			':login' => $login,
			':password' => $password
		));

		$data = $sth->fetch();

		$count = $sth->rowCount();

		if($count > 0){
			$perfil = $data['perfil'];
			$user_id = $data['user_id'];

			if($perfil == 'cliente'){
				$sth = $this->db->prepare("SELECT login FROM
usuarios WHERE mesa = :mesa AND user_id <> :user_id");
				$sth->execute(array(
					':mesa' => $mesa,
					':user_id' => $user_id
				));
				$data2 = $sth->fetch();
				$count2 = $sth->rowCount();

				if($count2 > 0){
					return '1';
				}
				else{
					$sth = $this->db->prepare("UPDATE usuarios SET
mesa = :mesa WHERE user_id = :user_id");
					$sth->execute(array(
						':mesa' => $mesa,
						':user_id' => $user_id
					));
				}
			}

			Session::init();
			Session::set('perfil', $perfil);
			Session::set('user_id', $user_id);
			Session::set('loggedIn',true);

			switch($perfil){

				case 'cliente':
					header('location:'.URL.'index');
					break;

				case 'admin':
					header('location:'.URL.'user');
```

```
                        break;

                    case 'cozinha':
                        header('location:'.URL.'listaPedidos');
                        break;

                    case 'caixa':
                        header('location:'.URL.'listaContas');
                        break;

                }

            }
            else{
                return '0';
            }

        }

        public function cadastrar(){

            $login = $_POST['email'];
            $password = $_POST['password'];
            $confirm_password = $_POST['confirm-password'];

            if($password == $confirm_password){

                $sth = $this->db->prepare("SELECT user_id FROM usuarios
WHERE login = :login");
                $sth->execute(array(
                    ':login' => $login,
                ));

                $count = $sth->rowCount();

                if($count == 0){
                    $sth = $this->db->prepare("INSERT INTO usuarios
(login, password) VALUES (:login, SHA1(:password))");
                    $sth->execute(array(
                        ':login' => $login,
                        ':password' => $password
                    ));

                    echo '1';
                }
                else{
                    echo '0';
                }
            }
        }

        public function recuperarSenha(){

            $login = $_POST['email'];

            $sth = $this->db->prepare("SELECT user_id FROM usuarios WHERE
login = :login");
            $sth->execute(array(
                                            ':login' => $login,
                                    ));
```

```
            $data = $sth->fetch();

            $count = $sth->rowCount();

            if($count == 1){

                  $nova_senha = '';
                  $caracteres =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890@';

                  $tamanho = strlen($caracteres);

                  for ($n = 1; $n <= 10; $n++) {
                        $numero_aleatorio = mt_rand(1, $tamanho);
                        $nova_senha .= $caracteres[$numero_aleatorio-1];
                  }

                  $sth = $this->db->prepare("UPDATE usuarios SET password =
SHA1(:password) WHERE login = :login");
                  $sth->execute(array(
                                                ':password' => $nova_senha,
                                                ':login' => $login,
                                          ));

                  $to = $login;
                  $from = "marco.santos.bsb@gmail.com";

                  $subject = "Nova senha";
                  $message = "Este e um e-mail automatico, favor nao
responder. \n\nSua nova senha e: " .$nova_senha. ".\n\nPara altera-la,
basta acessar o sistema e clicar em \"Alterar senha\".";
                  $envio = mail($to, $subject, $message, 'From: '.$from);

                  if($envio){
                        echo '1';
                  }
                  else{
                        echo '2';
                  }

            }
            else {
                  echo '0';
            }

      }

}
```

- logout_model.php:

```
<?php

class logout_model extends Model {

      public function __construct(){
            parent::__construct();
      }
```

```
	public function logout(){
		Session::destroy();
		header('location:'.URL.'login');
	}

}
```

- user_model.php:

```
<?php

class user_model extends Model {

	public function __construct(){
		parent::__construct();
	}

	public function userList(){
		$sth = $this->db->prepare('SELECT user_id,login,perfil FROM
usuarios WHERE perfil = "cozinha" OR perfil = "caixa"');
		$sth->execute();
		return $sth->fetchAll();
	}

	public function userSingleList($user_id){
		$sth = $this->db->prepare('SELECT user_id,login,perfil FROM
usuarios where user_id = :user_id');
		$sth->execute(array(':user_id' => $user_id));
		return $sth->fetch();
	}

	public function create($data){

		$sth = $this->db->prepare('INSERT INTO usuarios
(login,password,perfil) VALUES

	(:login,SHA1(:password),:perfil)');
		$sth->execute(array(
					':login' => $data['login'],
					':password' => $data['password'],
					':perfil' => $data['perfil']
					));
	}

	public function editSave($data){

		$sth = $this->db->prepare('UPDATE usuarios SET
							login = :login, password =
SHA1(:password), perfil = :perfil
							WHERE user_id =
:user_id');
		$sth->execute(array(
					':user_id' => $data['user_id'],
					':login' => $data['login'],
					':password' => $data['password'],
					':perfil' => $data['perfil']
					));
	}
```

```
	public function delete($user_id){
		$sth = $this->db->prepare('DELETE FROM usuarios WHERE user_id =
:user_id limit 1');
		$sth->execute(array(
					':user_id' => $user_id
					));
	}


}
```

Pasta “public”,”css”:

- default.css:

```
body {
	padding: 0;
	margin: 0;
	overflow-y: scroll;
	font: 18px arial, 'trebuchet ms', sans-serif;
}

/** Structure */

#nav {
	background-color: #222;
	font-size: 22px;
}

#content {
	width: 960px;
	margin: 0 auto;
	padding: 20px;
	text-align: left;
	background: #fff;
}

/* #footer {
	background: #12407f;
	color: #fff;
	padding: 20px;
	border-top: 4px solid #436fac;
	border-bottom-left-radius: 10px;
	border-bottom-right-radius: 10px;
} */

/* Links */

#nav_wrapper {
	width: 1000px;
	margin: 0 auto;
	text-align: left;
}

#nav ul {
	list-style-type: none;
```

```
      padding: 0;
      margin: 0;
      position: relative;
}

#nav ul li {
      display: inline-block;
}

#nav ul li:hover {
      background-color: #333;
}

#nav ul li a,visited {
      color: #ccc;
      display: block;
      padding: 15px;
      text-decoration: none;
}

#nav ul li a:hover {
      color: #ccc;
      text-decoration: none;
}

/* Conteúdo oculto */

#mask {
 background: url("/restaurante/public/images/login/bg.png");
 width: 100%;
 height: 100%;
 position: fixed!important;
 position: absolute;
 z-index: 9000;
 display: none;
}

.oculto, .avaliacao {
 position: absolute;
 display: none;
 z-index: 9999;
}
```

Pasta “views”:

- header.php:

```
<!doctype html>
<html>
<head>
      <title>Restaurante</title>
      <meta http-equiv="Content-Type" content="text/html;charset=iso-8859-
1" >
      <link rel="stylesheet" href="<?php echo URL;
?>public/css/default.css" >
      <script type="text/javascript" src="<?php echo URL;
?>public/js/jquery.js"></script>
      <?php
            if (isset($this->js)){
                  foreach ($this->js as $js){
```

```
                        echo '<script type="text/javascript"
src="'.URL.'views/'.$js.'"></script>';
                  }
            }
            if (isset($this->css)){
                        echo '<link rel="stylesheet"
href="'.URL.'views/'.$this->css.'" >';
            }
      ?>
</head>
<body>

<?php Session::init(); ?>

<!-- Div que irá escurecer a tela de fundo -->
<div id="mask"></div>

<!-- Div oculta de pedido que será chamada pela function
openModalFormPedido -->

<div class="oculto">

      <div id="escolher-pedido">

            <div id="detalhes-pedido">

            </div>

            <form action="<?php echo URL;?>index/cadastrarPedido"
method="post" id="form-pedido">

                  <div id="observacao">
                        <input type="text" size="61" maxlength="70"
class="input-text" title="Observações..." id="adicionar-observacao"
name="observacao">
                  </div>

                  <div id="enviar-pedido">

                        <select name="quantidade" id="quantidade">
                          <option value="1">1</option>
                          <option value="2">2</option>
                          <option value="3">3</option>
                          <option value="4">4</option>
                          <option value="5">5</option>
                          <option value="6">6</option>
                          <option value="7">7</option>
                        </select>

                        Total: R$<span id="total"></span>

                        <input type="submit" value="Adicionar"
id="adicionar" class="right">
                  </div>

            </form>

      </div>

</div>
```

```
<!-- Div oculta de cadastro que será chamada pela function
openModalFormAvaliacao -->

<div class="avaliacao">

      <div id="avaliar">

            <form action="<?php echo URL;?>index/fecharConta" method="post"
id="form-fechar">
                  <p>
                        Por favor, avalie o atendimento:
                              <select name="avaliacao">
                                <option value="Péssimo">Péssimo</option>
                                <option value="Fraco">Fraco</option>
                                <option value="Regular">Regular</option>
                                <option value="Bom">Bom</option>
                                <option value="Ótimo"
selected>Ótimo</option>
                              </select>
                  </p>
                  <br>
                  <p>
                        Comentários: <textarea id="comentario"
name="comentario" rows="4" cols="53" maxlength="400" ></textarea>
                  </p>
                  <p>
                        <input type="submit" value="Enviar" id="submit-
fechar" >
                  </p>
            </form>

      </div>

</div>


<div id="nav">
      <div id="nav_wrapper">

            <ul>

            <?php if (Session::get('loggedIn') == true){?>

                  <?php if (Session::get('perfil') == 'cliente'){?>
                        <li><a href="<?php echo URL;
?>index">Cardápio</a></li>
                        <li><a href="<?php echo URL;
?>duvidas">Dúvidas</a></li>
                        <li><a href="<?php echo URL; ?>alteraSenha">Alterar
senha</a></li>

                        <?php }else if (Session::get('perfil') ==
'admin'){?>
                              <li><a href="<?php echo URL;
?>user">Administrar usuários</a></li>
                              <li><a href="<?php echo URL;
?>avaliacoes">Avaliações</a></li>

                              <?php }else if(Session::get('perfil') ==
'cozinha'){?>
```

```
                              <li><a href="<?php echo URL;
?>listaPedidos">Lista de pedidos</a></li>
                              <li><a href="<?php echo URL;
?>listaDuvidas">Lista de dúvidas</a></li>
                              <li><a href="<?php echo URL;
?>cardapio">Cardápio</a></li>

                                    <?php }else if(Session::get('perfil') ==
'caixa'){ ?>
                                          <li><a href="<?php echo URL;
?>listaContas">Contas para fechamento</a></li>
                                    <?php } ?>
            <li><a href="<?php echo URL; ?>logout">Logout</a></li>
            <?php } ?>

            </ul>

      </div>
</div>

<div id="content">
```

- footer.php:

```
</div>

<div id="footer">

</div>

</body>

</html>
```

Pasta “views”,”alteraSenha”:

- index.php:

```
<div id="mensagem" class="erro"></div>

<h2>Alterar senha</h2>

<form id="change-password" method="post" action="<?php echo URL;
?>alteraSenha/alterar">
      <table>

      <tr>
            <td class="alinhar">Senha atual</td>
            <td><input type="password" id="current-password" name="current-
password" required></td>
      </tr>

      <tr>
            <td class="alinhar">Nova senha</td>
            <td><input type="password" id="new-password" name="new-
password" required></td>
      </tr>
```

```
	<tr>
		<td class="alinhar">Confirmar nova senha</td>
		<td><input type="password" id="confirm-password" name="confirm-
password" required></td>
	</tr>

	<tr>
		<td></td>
		<td><input type="submit" value="Enviar" /></td>
	</tr>

	</table>

</form>
```

Pasta “views”,”alteraSenha”,”css”:

- style.css:

```
/* Forms */

.alinhar {
	width: 180px;
	padding: 0px;
	margin: 10px 0px 0px;
	text-align: right;
}

form input, form select {
	width: auto;
	margin: 5px 0 0 10px;
	padding: 4px;
}

/* Mensagens de validação */

.erro {
	color: red;
	font-size: 17px;
	text-align: left;
	width: 500px;
	margin: 15px;
}

.confirmacao {
	color: blue;
}

#mensagem {
	font-size: 20px;
}
```

Pasta “views”,”alteraSenha”,”js”:

- default.js:

```
$(document).ready(function(){
```

```
      $("#change-password").submit(function(){
            var current_password = $("#current-password").val();
            var password = $("#new-password").val();
            var confirm_password = $("#confirm-password").val();
            var tamanhoSenha = password.length;

            if(tamanhoSenha >= 6){
              if(password == confirm_password){
                    var url = $(this).attr('action');
                    var data = $(this).serialize();

                    $.post(url, data, function(o) {
                          if(o == '1'){
                                $('#mensagem').addClass('confirmacao');
                                $('#mensagem').html('Senha alterada com
sucesso');
                                $("#current-password").val('');
                                $("#new-password").val('');
                                $("#confirm-password").val('');
                          }
                          else{
                                $('#mensagem').removeClass('confirmacao');
                                $('#mensagem').html('A senha atual está
incorreta');
                          }
                    });
              }
              else{
                    $('#mensagem').removeClass('confirmacao');
                    $('#mensagem').html('As senhas informadas não
coincidem');
              }
            }
            else{
              $('#mensagem').removeClass('confirmacao');
              $('#mensagem').html('A nova senha deve ter no mínimo 6
caraceteres');
            }

            return false;
         });

});
```

Pasta “views”,”avaliacoes”:

- index.php:

```
<h2>Lista de avaliações</h2>

<table id="listaDeAvaliacoes">

<?php
      if(!empty($this->avaliacoesList)){
            echo '<tr>';
            echo '<td class="bold">Avaliação</td><td
class="bold">Comentário</td>';
            echo '</tr>';
```

```
            foreach($this->avaliacoesList as $key => $value){
                  echo '<tr>';
                  echo '<td>'.$value['avaliacao'].'</td>';
                  echo '<td>'.$value['comentario'].'</td>';
                  echo '</tr>';
            }
      }
      else{
            echo '<p>Não existem avaliações a serem exibidas.</p>';
      }
?>
</table>
```

Pasta “views”,”avaliacoes”,”css”:

- style.css:

```
#listaDeAvaliacoes {
      border-collapse: collapse;
}

td {
      border: 1px solid black;
      vertical-align: top;
      padding: 5px;
}

.bold {
      font-weight: bold;
}
```

Pasta “views”,”cardapio”:

- index.php:

```
<h2>Cardápio: Adicionar item</h2>


<form method="post" action="<?php echo URL; ?>cardapio/adicionar/">
      <table>

      <tr>
            <td class="alinhar">Categoria</td>
            <td>
                  <select name="categoria">
                        <option value="bebida">Bebida</option>
                        <option value="prato">Prato</option>
                        <option
value="acompanhamento">Acompanhamento</option>
                        <option value="sobremesa">Sobremesa</option>
                  </select>
            </td>
      </tr>

      <tr>
            <td class="alinhar">Item</td>
            <td><input type="text" name="item" required></td>
      </tr>
```

```
	<tr>
		<td class="alinhar">Preço(R$)</td>
		<td><input type="text" name="preco" required></td>
	</tr>

	<tr>
		<td class="alinhar">Descrição</td>
		<td><input type="text" name="descricao"></td>
	</tr>

	<tr>
		<td></td>
		<td><input type="submit" value="Enviar" /></td>
	</tr>

	</table>

</form>
```

Pasta “views”,”cardapio”,”css”:

- style.css:

```
/** Forms */

.alinhar {
	width: 160px;
	padding: 0px;
	margin: 10px 0px 0px;
	text-align: right;
}

form input, form select {
	width: auto;
	margin: 5px 0 0 10px;
	padding: 4px;
}
```

Pasta “views”,”duvidas”:

- index.php:

```
<div id="msg"></div>

<p>Envie suas dúvidas:</p>

<form id="form-duvida" action="<?php echo URL;?>duvidas/duvidaInsert/"
method="post">
	<textarea id="duvida" name="duvida" rows="10" cols="60"
maxlength="500" required autofocus></textarea><br>
	<input type="reset" value="Limpar">
	<input type="hidden" name="user_id" value="<?php echo
Session::get('user_id'); ?>">
	<input type="submit" name="submit" value="Enviar">
</form>
<br>
<hr>

<h2>Respostas</h2>
```

```
<div id="respostas"></div>
```

Pasta “views”,”duvidas”,”css”:

- style.css:

```
#msg {
	color:blue;
	font-size: 17px;
}
```

Pasta “views”,”duvidas”,”js”:

- default.js:

```
//ultimo ID que foi buscado no banco de dados
var lastID = 0;
var showMsg = true;

$(document).ready(function(){
	atualiza();

 });

var atualiza = function(){

	$.get('duvidas/respostasList', function(o) {

		if(o.length < 1 && showMsg){
			$('#respostas').empty();
			$('#respostas').append('<div>Nenhuma dúvida ainda feita
ou respondida.</div>');
			showMsg = false;
			return false;
		}

		if(o.length >= 1){

			if(showMsg == false){
				$('#respostas').empty();
				showMsg = true;
			}

			//ultima linha do JSON
			var lastLine = o.length - 1;

			if(lastID == 0){
				for (var i = 0; i < o.length; i++){
					$('#respostas').append('<div><b>Dúvida: "' +
o[i].duvida + '"</b><br>Resposta: "' + o[i].resposta + '"</div><br>');
				}
				lastID = o[lastLine].duvida_id;
			}
			else{
				//se o último ID da busca anterior for menor do que
o último ID da busca atual, inserir os novos dados
				if(lastID < o[lastLine].duvida_id){

					//lastID = o[lastLine].id;
```

```
                                    for (var i = 0; i < o.length; i++){
                                          if(o[i].duvida_id > lastID ){

      $('#respostas').append('<div><b>Dúvida: "' + o[i].duvida +
'"</b><br>Resposta: "' + o[i].resposta + '"</div><br>');
                                                lastID = o[i].duvida_id;
                                    }
                                }
                           }
                     }
               }

      },'json');

//chama "atualiza" a cada 5 segundos (5000 ms)
setTimeout('atualiza()', 5000);
}
```

- validaEnvio.js:

```
$(document).ready(function(){

      $("#form-duvida").submit(function(){

            var url = $(this).attr('action');
          var data = $(this).serialize();

          $.post(url, data, function(o) {
            $('#duvida').val('');
            $('#msg').empty();
            $('#msg').append('Sua dúvida foi enviada');
          });

          return false;

      });

});
```

Pasta “views”,”erro”:

- index.php:

```
<?php echo $this->msg; ?>
```

Pasta “views”,”index”:

- index.php:

```
<div id="pedidos">
      <p class="bold center">Meus pedidos</p>

      <hr>

      <div id="lista-pedidos">

            <table id="tabela-pedidos">

            </table>
```

```
      </div>

      <hr>

      <p>Subtotal: R$<span id="subtotal">0.00</span></p>
      <p>
            Adicional 10% garçom:
            <select name="adicional-garçom" id="adicional-garçom">
              <option value="sim">Sim</option>
              <option value="nao">Não</option>
            </select>
      </p>

      <hr>

      <div id="solicitar-fechamento">
            <p>Total: R$<span id="valor-final">0.00</span> <a
href="javascript:openModalFormAvaliacao();"><button type="button"
id="fechar-conta">Fechar conta</button></a><p>
      </div>

</div>

<div id="cardapio">

      <div id="bebidas">
            <h2>Bebidas</h2>
      </div>

      <br class="clearBoth" />

      <div id="pratos">
            <h2>Pratos</h2>
      </div>

      <br class="clearBoth" />

      <div id="acompanhamentos">
            <h2>Acompanhamentos</h2>
      </div>

      <br class="clearBoth" />

      <div id="sobremesas">
            <h2>Sobremesas</h2>
      </div>

      <br class="clearBoth" />

</div>
```

Pasta “views”,”index”,”css”:

- style.css:

```
h2 {
      font-size: 23px;
}

p {
```

```
	margin: 0px;
	padding: 0px;
	font-size: 18px;
}

button {
	padding: 5px 8px;
}

#cardapio {
	float: left;
	width:650px;
}

.item-cardapio {
	float:left;
	width: 309px;
	color: black;
	margin: 4px;
	padding: 4px;
	background-color: #E6E6FA;
}

.clearBoth {
	clear:both;
}

.right {
	float:right;
	text-align: right;
	margin-left: 15px;
}

.left {
	float:left;
	text-align: left;
	margin-bottom: 5px;
}

.descricao {
	font-size: 16px;
	color: #555;
}

#escolher-pedido, #avaliar {
	 margin: 0 auto;
	 width: 400px;
	 min-height: 100px;
	 display: table;
	 background: #fff;
	 border-radius: 5px;
	 border: 1px solid #f90;
	 padding: 20px;
	 margin-bottom: 50px;
}

.lightcolor {
	color: #888;
}

#escolher-pedido input {
```

```
	padding: 7px;
}

#detalhes-pedido {
	margin-bottom: 15px;
}

#observacao {
	margin-bottom: 15px;
}

#enviar-pedido select {
	padding: 7px 12px;
}

#avaliar select {
	padding: 5px 7px;
}

#enviar-pedido {
	display: inline;
}

#pedidos {
	margin-top: 30px;
	padding: 5px;
	float: right;
	width: 290px;
	border: 1px solid #E6E6FA;
	background-color: #ddd;
}

.bold {
	font-weight: bold;
	font-size: 17px;
}

.center {
	text-align: center;
}

#pedidos > p, #lista-pedidos > p, tr {
	font-size: 17px;
}

td {
	vertical-align: top;
	padding: 5px;
}

#submit-fechar {
	padding: 7px;
}

#msg-fechamento {
	background-color: purple;
	color: white;
	text-align: center;
}

#msgContaFechada {
```

```
	text-align: center;
	color: red;
	padding-bottom: 5px;
}
```

Pasta “views”,”index”,”js”:

- default.js:

```
$(document).ready(function(){

	var subtotal = 0;
	var valor_final = 0;
	var adicional = $('#adicional-garçom').val();

	//Ao carregar a página, buscar os itens do cardápio e inserí-los
dentro das respectivas categorias.
	$.get('index/getCardapio', function(o) {

		var qtdBebidas = 0;
		var qtdPratos = 0;
		var qtdAcompanhamentos = 0;
		var qtdSobremesas = 0;

		for (var i = 0; i < o.length; i++){

			switch(o[i].categoria) {

			    case 'bebida':
			      if((qtdBebidas % 2 == 0) && (qtdBebidas != 0)){
					$('#bebidas').append('<br class="clearBoth"
/>');
			      }
			      $('#bebidas').append('<a
href="javascript:openModalFormPedido('+ o[i].item_id +');"><div
class="item-cardapio"><p class="left">'+ o[i].item +'</p><p
class="right">R$'+ o[i].preco +'</p><p class="clearBoth descricao">'+
o[i].descricao +'</p></div></a>');
			      qtdBebidas ++;
			      break;

			    case 'prato':
			      if((qtdPratos % 2 == 0) && (qtdPratos != 0)){
					$('#pratos').append('<br class="clearBoth"
/>');
			      }
			      $('#pratos').append('<a
href="javascript:openModalFormPedido('+ o[i].item_id +');"><div
class="item-cardapio"><p class="left">'+ o[i].item +'</p><p
class="right">R$'+ o[i].preco +'</p><p class="clearBoth descricao">'+
o[i].descricao +'</p></div></a>');
			      qtdPratos ++;
			      break;

			    case 'acompanhamento':
			      if((qtdAcompanhamentos % 2 == 0) &&
(qtdAcompanhamentos != 0)){
					$('#acompanhamentos').append('<br
class="clearBoth" />');
			      }
```

```
                        $('#acompanhamentos').append('<a
href="javascript:openModalFormPedido('+ o[i].item_id +');"><div
class="item-cardapio"><p class="left">'+ o[i].item +'</p><p
class="right">R$'+ o[i].preco +'</p><p class="clearBoth descricao">'+
o[i].descricao +'</p></div></a>');
                        qtdAcompanhamentos ++;
                        break;

                      case 'sobremesa':
                        if((qtdSobremesas % 2 == 0) && (qtdSobremesas !=
0)){
                              $('#sobremesas').append('<br class="clearBoth"
/>');
                        }
                        $('#sobremesas').append('<a
href="javascript:openModalFormPedido('+ o[i].item_id +');"><div
class="item-cardapio"><p class="left">'+ o[i].item +'</p><p
class="right">R$'+ o[i].preco +'</p><p class="clearBoth descricao">'+
o[i].descricao +'</p></div></a>');
                        qtdSobremesas ++;
                        break;

                  }

            }

      },'json');

      //Ao carregar a página, buscar os pedidos do usuário. Se o usuário
ainda não fez pedidos, mostrar mensagem.
      $.get('index/getPedidos', function(o) {

            if(o == '0') {
                  $('#lista-pedidos').append('<p id="nenhum-pedido">Você
ainda não solicitou nenhum pedido</p>');
                  $('#fechar-conta').hide();
            }
            else {
                  $('#tabela-pedidos').append('<tr><td
class="bold">Qtd.</td><td class="bold">Pedido</td><td
class="bold">Total</td></tr>');

                  for (var i = 0; i < o.length; i++){
                        $('#tabela-pedidos').append('<tr><td
class="center">'+ o[i].quantidade +'</td><td>'+ o[i].item +'</td><td>'+
o[i].valor_total +'</td></tr>');
                        subtotal = parseFloat(o[i].valor_total) +
parseFloat(subtotal);
                        subtotal = subtotal.toFixed(2);
                  }
                  $('#subtotal').html(subtotal);

                  valor_final = subtotal*1.1;
                  valor_final = valor_final.toFixed(2);
                  $('#valor-final').html(valor_final);
            }

      },'json');

      //Ao carregar a página, verificar se o usuário solicitou o fechamento
da conta.
```

```
	$.get('index/getStatusConta', function(o) {
		if(o == '1') {
			$('#fechar-conta').hide();
			$('#solicitar-fechamento').append('<br><p id="msg-
fechamento">Sua conta será fechada em instantes</p>');
		}
	},'json');

	//Coloca um texto em fonte clara dentro da tag de observações para
informar o que o usuário deve inserir naquele campo
	$('.input-text').addClass('lightcolor');
	var inputtitle = $('.input-text').attr('title');
	$('.input-text').val(inputtitle);

	$('.input-text').live('focus', function(){
		if($(this).val() == $(this).attr('title')){
			$(this).val('').removeClass('lightcolor');
		}
	});

	$('.input-text').live('blur', function(){
		if($(this).val() == ''){
			$(this).val(inputtitle).addClass('lightcolor');
		}
	});

	//Atualiza o valor total do pedido sempre que o usuário alterar a
quantidade do item
	$('#quantidade').live('blur', function(){
		var quantidade = $(this).val();
		var itemPreco = $('#valor-item').text();
		var total = quantidade*itemPreco;
		total = total.toFixed(2);
		$('#total').html(total);
	});

	//Atualiza o valor total se o usuário colocar ou retirar o adicional
do garçom
	$('#adicional-garçom').live('blur', function(){
		adicional = $('#adicional-garçom').val();
		if(adicional == 'sim'){
			valor_final = subtotal*1.1;
			valor_final = valor_final.toFixed(2);
			$('#valor-final').html(valor_final);
		}
		else {
			valor_final = subtotal;
			$('#valor-final').html(valor_final);
		}
	});

	//Atualiza a lista de pedidos após adicionar o novo pedido
	$("#form-pedido").submit(function(){
		var url = $(this).attr('action');
		var data = $(this).serialize();

		//Verifica se este é o primeiro pedido do usuário, checando se
a tag com id "nenhum-pedido" está no html
		//Se for o primeiro pedido, remove a tag com o texto que
informa que o usuário ainda não fez pedidos
		if($("#nenhum-pedido").length){
```

```
                  $('#fechar-conta').show();
                  $("#nenhum-pedido").remove();
                  $('#tabela-pedidos').append('<tr><td
class="bold">Qtd.</td><td class="bold">Pedido</td><td
class="bold">Total</td></tr>');
            }

            $.post(url, data, function(o){

                  if(o == 1){
                        $('#escolher-pedido').prepend('<p
id="msgContaFechada">Você já solicitou o fechamento da conta</p>');
                  }
                  else{

                        var valor_total = parseFloat(o.valor_total);
                        valor_total = valor_total.toFixed(2);
                        $('#tabela-pedidos').append('<tr><td
class="center">'+ o.quantidade +'</td><td>'+ o.item +'</td><td>'+
valor_total +'</td></tr>');

                        subtotal = $('#subtotal').html();
                        subtotal = parseFloat(subtotal) +
parseFloat(valor_total);
                        subtotal = subtotal.toFixed(2);
                        $('#subtotal').html(subtotal);

                        if(adicional == 'sim'){
                              valor_final = subtotal*1.1;
                              valor_final = valor_final.toFixed(2);
                              $('#valor-final').html(valor_final);
                        }
                        else {
                              valor_final = subtotal
                              $('#valor-final').html(valor_final);
                        }

                        $('#mask').hide();
                        $('.oculto').hide();
                        $('#detalhes-pedido').empty();
                        $('#total').empty();
                        $('#item-id').remove();
                        $('#item').remove();
                        $('#item-valor').remove();
                        $('#quantidade').val('1');

                        $('.input-text').addClass('lightcolor');
                        var inputtitle = $('.input-text').attr('title');
                        $('.input-text').val(inputtitle);
                  }
            },'json');

            return false;
      });

});
```

- mostrarFormAvaliacao.js:

```
function openModalFormAvaliacao() {
```

```
	var total_conta = $('#valor-final').html();
	$('#form-fechar').append('<input type="hidden" name="total" value="'+
total_conta +'">');

	$('#mask').fadeIn(500);
	$('#mask').fadeTo("slow",0.5);

	var h = $(document).height();
	var w = $(document).width();

	var fh = h / 2 - $('.avaliacao').height() / 2;
	var fw = w / 2 - $('.avaliacao').width() / 2;

	$('.avaliacao').css('top', fh);
	$('.avaliacao').css('left', fw);

	$('.avaliacao').fadeIn(1000);

	$('#mask').click(function() {
		$('#mask').hide();
		$('.avaliacao').hide();
	});

}
```

- mostrarItem.js:

```
function openModalFormPedido(item_id) {

	//traz as informações do item selecionado
	$.post('index/getItem',{item_id:item_id}, function(o) {
		$('#detalhes-pedido').prepend('<div><p class="left">'+ o.item
+'</p><p class="right">R$<span id="valor-item">'+ o.preco +'</span></p><p
class="clearBoth descricao">'+ o.descricao +'</p></div>');
		$('#total').append(o.preco);
		$('#form-pedido').prepend('<input type="hidden" name="item-id"
id="item-id" value="'+ o.item_id +'">');
		$('#form-pedido').prepend('<input type="hidden" name="item"
id="item" value="'+ o.item +'">');
		$('#form-pedido').prepend('<input type="hidden" name="item-
valor" id="item-valor" value="'+ o.preco +'">');
	},'json');

	$('#mask').fadeIn(500);
	$('#mask').fadeTo("slow",0.5);

	var h = $(document).height();
	var w = $(document).width();

	var fh = h / 2 - $('.oculto').height() / 2;
	var fw = w / 2 - $('.oculto').width() / 2;

	$('.oculto').css('top', fh);
	$('.oculto').css('left', fw);

	$('.oculto').fadeIn(1000);

	$('#mask').click(function() {
		$('#mask').hide();
		$('.oculto').hide();
		$('#detalhes-pedido').empty();
```

```
            $('#total').empty();
            $('#msgContaFechada').remove();
            $('#item-id').remove();
            $('#item').remove();
            $('#item-valor').remove();
            $('#quantidade').val('1');


            $('.input-text').addClass('lightcolor');
            var inputtitle = $('.input-text').attr('title');
            $('.input-text').val(inputtitle);
      });

}
```

Pasta “views”,”listaContas”:

- index.php:

```
<h2>Contas aguardando fechamento</h2>

<div id="listaDeContas">

</div>
```

Pasta “views”,”listaContas”,”css”:

- style.css;

```
#listaDeContas {
      border-collapse: collapse;
}

#fechar {
      border: none;
}

td {
      border: 1px solid black;
      vertical-align: top;
      padding: 5px;
}

.bold {
      font-weight: bold;
}

.center {
      text-align: center;
}
```

Pasta “views”,”listaContas”,”js”:

- default.js:

```
//ultimo ID que foi buscado no banco de dados
var lastID = 0;
var showMsg = true;

$(document).ready(function(){
      atualiza();

});

var atualiza = function(){

      $.get('listaContas/contasList', function(o) {

            if(o.length < 1 && showMsg){
                  $('#listaDeContas').empty();
                  $('#listaDeContas').append('<div>Nenhuma conta aguardando
fechamento.</div>');
                  showMsg = false;
                  lastID = 0;
                  return false;
            }

            if(o.length >= 1){

                  if(showMsg == false){
                        $('#listaDeContas').empty();
                        showMsg = true;
                  }

                  //ultima linha do JSON
                  var lastLine = o.length - 1;

                  if(lastID == 0){

                        $('#listaDeContas').append('<tr
class="bold"><td>Mesa</td><td>Login</td><td>Valor</td></tr>');

                        $('#listaDeContas').append('<tr><td
class="center">' + o[0].mesa + '</td>' +
                                                                  '<td>' +
o[0].login + '</td>' +
                                                                  '<td
class="center">' + o[0].valor + '</td>' +
                                                                  '<td
id="fechar"><a href="listaContas/fecharPedido/' + o[0].conta_id +
'">Fechar</a></td></tr>');

                        lastID = o[0].conta_id;

                        for (var i = 1; i < o.length; i++){
                              $('#listaDeContas').append('<tr><td
class="center">' + o[i].mesa + '</td>' +
                                                                        '<td>'
+ o[i].login + '</td>' +
                                                                        '<td
class="center">' + o[i].valor + '</td></tr>');
                        }
                        lastID = o[lastLine].conta_id;
                  }
                  else{
```

```
                        //se o último ID da busca anterior for menor do que
o último ID da busca atual, inserir os novos dados
                        if(lastID < o[lastLine].conta_id){

                              for (var i = 1; i <= o.length; i++){
                                    if(o[i].conta_id > lastID ){

      $('#listaDeContas').append('<tr><td class="center">' + o[i].mesa +
'</td>' +
                                                                  '<td>' + o[i].login
+ '</td>' +
                                                                  '<td
class="center">' + o[i].valor + '</td></tr>');
                                                lastID = o[i].conta_id;
                                    }
                              }
                        }
                  }
            }

      },'json');

//chama "atualiza" a cada 10 segundos (10000 ms)
setTimeout('atualiza()', 10000);
}
```

Pasta “views”,”listaDuvidas”:

- index.php:

```
<h2>Lista de dúvidas</h2>

<div id="listaDeDuvidas">

</div>
```

- response.php:

```
<h2>Responder dúvida</h2>

<?php echo '"'.$this->duvida['duvida']. '"';?>

<br><br>

<form method="post" action="<?php echo URL;
?>listaDuvidas/responseSave/<?php echo $this->duvida['duvida_id'];?>">
      <textarea id="resposta" name="resposta" rows="10" cols="60"
maxlength="500" required autofocus></textarea><br>
      <input type="reset" value="limpar">
      <input type="submit" name="submit" value="Enviar">
</form>
```

Pasta “views”,”listaDuvidas”,”css”:

- style.css:

```
a {
	text-decoration: none;
}
```

Pasta “views”,”listaDuvidas”,”js”:

- default.js:

```
//ultimo ID que foi buscado no banco de dados
var lastID = 0;
var showMsg = true;

$(document).ready(function(){
	atualiza();

});

var atualiza = function(){

	$.get('listaDuvidas/duvidasList', function(o) {

		if(o.length < 1 && showMsg){
			$('#listaDeDuvidas').empty();
			$('#listaDeDuvidas').append('<div>Nenhuma dúvida
aguardando resposta.</div>');
			showMsg = false;
			lastID = 0;
			return false;
		}

		if(o.length >= 1){

			if(showMsg == false){
				$('#listaDeDuvidas').empty();
				showMsg = true;
			}

			//ultima linha do JSON
			var lastLine = o.length - 1;

			if(lastID == 0){

				$('#listaDeDuvidas').append('<div>"' + o[0].duvida
+
					'"<br><a href="listaDuvidas/response/' +
o[0].duvida_id + '">Responder</a>' +
					'<a href="listaDuvidas/delete/' +
o[0].duvida_id + '"> Deletar</a><br><br></div>');

				lastID = o[0].duvida_id;

				for (var i = 1; i < o.length; i++){
					$('#listaDeDuvidas').append('<div>"' +
o[i].duvida + '"<br><br></div>');
				}
				lastID = o[lastLine].duvida_id;
			}
			else{
```

```
                        //se o último ID da busca anterior for menor do que
o último ID da busca atual, inserir os novos dados
                        if(lastID < o[lastLine].duvida_id){

                              for (var i = 1; i <= o.length; i++){
                                    if(o[i].duvida_id > lastID ){

      $('#listaDeDuvidas').append('<div>"' + o[i].duvida +
'"<br><br></div>');
                                                lastID = o[i].duvida_id;
                                    }
                              }
                        }
                  }
            }

      },'json');

//chama "atualiza" a cada 3 segundos (5000 ms)
setTimeout('atualiza()', 3000);
}
```

Pasta “views”,”listaPedidos”:

- index.php:

```
<h2>Lista de pedidos</h2>

<div>
      <table id="listaDePedidos">

      </table>
</div>
```

Pasta “views”,”listaPedidos”,”css”:

- style.css:

```
#listaDePedidos {
      border-collapse: collapse;
}

#atender {
      border: none;
}

td {
      border: 1px solid black;
      vertical-align: top;
      padding: 5px;
}

.bold {
      font-weight: bold;
}
```

```
.center {
	text-align: center;
}

.obs {
	max-width: 350px;
}
```

Pasta “views”,”listaPedidos”,”js”:

- default.js:

```
//ultimo ID que foi buscado no banco de dados
var lastID = 0;
var showMsg = true;

$(document).ready(function(){
	atualiza();

});

var atualiza = function(){

	$.get('listaPedidos/pedidosList', function(o) {

		if(o.length < 1 && showMsg){
			$('#listaDePedidos').empty();
			$('#listaDePedidos').append('<div>Nenhum pedido
aguardando atendimento.</div>');
			showMsg = false;
			lastID = 0;
			return false;
		}

		if(o.length >= 1){

			if(showMsg == false){
				$('#listaDePedidos').empty();
				showMsg = true;
			}

			//ultima linha do JSON
			var lastLine = o.length - 1;

			if(lastID == 0){

				$('#listaDePedidos').append('<tr
class="bold"><td>Qtd.</td><td>Pedido</td><td>Mesa</td><td>Observações</td><
/tr>');

				$('#listaDePedidos').append('<tr><td
class="center">' + o[0].quantidade + '</td>' +
'<td>' +
o[0].item + '</td>' +
'<td
class="center">' + o[0].mesa + '</td>' +
```

```
                                                                        '<td
class="obs">' + o[0].observacao + '</td>' +
                                                                        '<td
id="atender"><a href="listaPedidos/atenderPedido/' + o[0].pedido_id +
'">Atender</a></td></tr>');

                        lastID = o[0].pedido_id;

                        for (var i = 1; i < o.length; i++){
                              $('#listaDePedidos').append('<tr><td
class="center">' + o[i].quantidade + '</td>' +

      '<td>' + o[i].item + '</td>' +
                                                                              '<td
class="center">' + o[i].mesa + '</td>' +
                                                                              '<td
class="obs">' + o[i].observacao + '</td></tr>');
                        }
                        lastID = o[lastLine].pedido_id;
                  }
                  else{
                        //se o último ID da busca anterior for menor do que
o último ID da busca atual, inserir os novos dados
                        if(lastID < o[lastLine].pedido_id){

                              for (var i = 1; i <= o.length; i++){
                                    if(o[i].pedido_id > lastID ){

      $('#listaDePedidos').append('<tr><td class="center">' +
o[i].quantidade + '</td>' +

            '<td>' + o[i].item + '</td>' +

            '<td class="center">' + o[i].mesa + '</td>' +

            '<td class="obs">' + o[i].observacao + '</td></tr>');
                                          lastID = o[i].pedido_id;
                                    }
                              }
                        }
                  }
            }

      },'json');

//chama "atualiza" a cada 10 segundos (10000 ms)
setTimeout('atualiza()', 10000);
}
```

Pasta “views”,”login”:

- index.php:

```
<!doctype html>
<html>

<head>
  <title>Login</title>
```

```
  <meta http-equiv="Content-Type" content="text/html;charset=iso-8859-1" />
  <link rel="stylesheet" href="<?php echo URL; ?>views/login/css/style.css"
/>
  <script type="text/javascript" src="<?php echo URL;
?>public/js/jquery.js"></script>
  <script type="text/javascript" src="<?php echo URL;
?>views/login/js/mostraForm.js"></script>
  <script type="text/javascript" src="<?php echo URL;
?>views/login/js/validar.js"></script>
</head>
<body>

      <!-- Div que irá escurecer a tela de fundo -->
      <div id="mask"></div>

      <!-- Div oculta de cadastro que será chamada pela function
openModalForm -->

      <div class="cadastro">

       <div class="form">

        <h2>Cadastro</h2>
        <div class="erro" id="mensagem-cadastro"></div>

         <form action="<?php echo URL;?>login/cadastrar" method="post"
id="form-cadastro">

          <p><label>E-mail:</label><input class="field" type="text"
size="32" name="email" id="mail1" required></p>
          <p><label>Senha:</label><input class="field" type="password"
size="30" id="senha1" name="password" required></p>
          <p><label>Confirmar senha:</label><input class="field"
type="password" size="30" id="senha2" name="confirm-password" required></p>
          <p><label></label><input name="submit" class="button"
type="submit" value="Enviar"></p>

         </form>

       </div>

      </div>

      <!-- Div oculta de recuperação de senha que será chamada pela
function openModalForm -->

      <div class="recuperar-senha">

       <div class="form">

        <h2>Recuperar senha</h2>
        <div class="erro" id="mensagem-recuperar"></div>

         <form action="<?php echo URL;?>login/recuperarSenha" method="post"
id="form-esqueci">

          <p><label>E-mail:</label><input class="field" type="text"
size="32" name="email" id="mail2" required></p>
          <p><label></label><input name="submit" class="button"
type="submit" value="Enviar" required></p>
```

```
         </form>

       </div>

      </div>

      <!-- Formulário de Login -->

      <?php if($this->logar == '0') echo '<div class="erro" id="mensagem-
login">O login ou senha digitado está incorreto</div>'; ?>
      <?php if($this->logar == '1') echo '<div class="erro" id="mensagem-
login">A mesa selecionada está ocupada. Favor escolher outra.</div>'; ?>

      <div id="titulo">
            Acesso ao sistema
      </div>

      <form method="post" action="<?php echo URL;?>login/run" id="form-
login" class="login">
            <p>
                  <label for="login">Login:</label>
                  <input type="text" name="login" id="login" required>
            </p>

      <p>
                  <label for="senha">Senha:</label>
                  <input type="password" name="password" id="password"
required>
            </p>

            <p>
                  <label for="mesa">Mesa:</label>
                  <select name="mesa" id="mesa">
                        <option value="1">1</option>
                        <option value="2">2</option>
                        <option value="3">3</option>
                        <option value="4">4</option>
                        <option value="5">5</option>
                        <option value="6">6</option>
                        <option value="7">7</option>
                        <option value="8">8</option>
                        <option value="9">9</option>
                        <option value="10">10</option>
                  </select>
            </p>

            <p class="login-submit">
                  <button type="submit" class="login-button">Login</button>
            </p>

            <p class="links"><a href="javascript:openModalForm('.recuperar-
senha');">Esqueci minha senha</a></p>
            <p class="links"><a
href="javascript:openModalForm('.cadastro');">Não possuo cadastro</a></p>
      </form>


</body>
</html>
```

Pasta “views”,”login”,”css”:

- style.css:

```
/* Configuração principal da página de login */

html, body, div, span, applet, object, iframe,
h1, h2, h3, h4, h5, h6, p, blockquote, pre,
a, abbr, acronym, address, big, cite, code,
del, dfn, em, img, ins, kbd, q, s, samp,
small, strike, strong, sub, sup, tt, var,
b, u, i, center,
dl, dt, dd, ol, ul, li,
fieldset, form, label, legend,
table, caption, tbody, tfoot, thead, tr, th, td,
article, aside, canvas, details, embed,
figure, figcaption, footer, header, hgroup,
menu, nav, output, ruby, section, summary,
time, mark, audio, video {
  margin: 0;
  padding: 0;
  border: 0;
  font-size: 100%;
  font: inherit;
  vertical-align: baseline;
}

article, aside, details, figcaption, figure,
footer, header, hgroup, menu, nav, section {
  display: block;
}

body {
  line-height: 1;
}

ol, ul {
  list-style: none;
}

blockquote, q {
  quotes: none;
}

blockquote:before, blockquote:after,
q:before, q:after {
  content: '';
  content: none;
}

table {
  border-collapse: collapse;
  border-spacing: 0;
}

/*
 * Copyright (c) 2012-2013 Thibaut Courouble
 * http://www.cssflow.com
 *
 * Licensed under the MIT License:
```

```
 * http://www.opensource.org/licenses/mit-license.php
 */

body, .login-submit, .login-submit:before, .login-submit:after {
  background: #373737 url("/restaurante/public/images/login/bg.png") 0 0
repeat;
}

body {
  font: 14px/20px 'Helvetica Neue', Helvetica, Arial, sans-serif;
  color: #404040;
}

a {
  color: #00a1d2;
  /* text-decoration: none; */
}
a:hover {
  text-decoration: underline;
}

.login {
  position: relative;
  margin: 50px auto;
  width: 400px;
  padding-right: 32px;
  font-weight: 300;
  color: #a8a7a8;
  text-shadow: 1px 1px 0 rgba(0, 0, 0, 0.8);
}
.login p {
  margin: 0 0 10px;
}

#mesa {
      padding: 2px 7px;
      margin-top: 9px;
}

input, button, label {
  font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif;
  font-size: 20px;
  font-weight: 300;
  -webkit-box-sizing: border-box;
  -moz-box-sizing: border-box;
  box-sizing: border-box;
}

input[type=text], input[type=password] {
  padding: 0 10px;
  width: 300px;
  height: 40px;
  color: #bbb;
  text-shadow: 1px 1px 1px black;
  background: rgba(0, 0, 0, 0.16);
  border: 0;
  border-radius: 5px;
  -webkit-box-shadow: inset 0 1px 4px rgba(0, 0, 0, 0.3), 0 1px rgba(255,
255, 255, 0.06);
  box-shadow: inset 0 1px 4px rgba(0, 0, 0, 0.3), 0 1px rgba(255, 255, 255,
0.06);
```

```
}

input[type=text]:focus, input[type=password]:focus {
  color: white;
  background: rgba(0, 0, 0, 0.1);
  outline: 0;
}

label {
  float: left;
  width: 100px;
  line-height: 40px;
  padding-right: 10px;
  font-weight: 100;
  text-align: right;
  letter-spacing: 1px;
}

.links {
  padding-left: 100px;
  padding-top:10px;
  font-size: 16px;
  font-weight: 100;
  letter-spacing: 1px;
}

.login-submit {
  position: absolute;
  top: 12px;
  right: 0;
  width: 48px;
  height: 48px;
  padding: 8px;
  border-radius: 32px;
  -webkit-box-shadow: 0 0 4px rgba(0, 0, 0, 0.35);
  box-shadow: 0 0 4px rgba(0, 0, 0, 0.35);
}
.login-submit:before, .login-submit:after {
  content: '';
  z-index: 1;
  position: absolute;
}
.login-submit:before {
  top: 28px;
  left: -4px;
  width: 4px;
  height: 10px;
  -webkit-box-shadow: inset 0 1px rgba(255, 255, 255, 0.06);
  box-shadow: inset 0 1px rgba(255, 255, 255, 0.06);
}
.login-submit:after {
  top: -4px;
  bottom: -4px;
  right: -4px;
  width: 36px;
}

.login-button {
  position: relative;
  z-index: 2;
  width: 48px;
```

```
  height: 48px;
  padding: 0 0 48px;
  /* Fix wrong positioning in Firefox 9 & older (bug 450418) */
  text-indent: 120%;
  white-space: nowrap;
  overflow: hidden;
  background: none;
  border: 0;
  border-radius: 24px;
  cursor: pointer;
  -webkit-box-shadow: inset 0 1px 1px rgba(0, 0, 0, 0.2), 0 1px rgba(255,
255, 255, 0.1);
  box-shadow: inset 0 1px 1px rgba(0, 0, 0, 0.2), 0 1px rgba(255, 255, 255,
0.1);
  /* Must use another pseudo element for the gradient background because
Webkit */
  /* clips the background incorrectly inside elements with a border-radius.
*/
}

.login-button:before {
  content: '';
  position: absolute;
  top: 5px;
  bottom: 5px;
  left: 5px;
  right: 5px;
  background: #00a2d3;
  border-radius: 24px;
  background-image: -webkit-linear-gradient(top, #00a2d3, #0d7796);
  background-image: -moz-linear-gradient(top, #00a2d3, #0d7796);
  background-image: -o-linear-gradient(top, #00a2d3, #0d7796);
  background-image: linear-gradient(to bottom, #00a2d3, #0d7796);
  -webkit-box-shadow: inset 0 0 0 1px #00a2d3, 0 0 0 5px rgba(0, 0, 0,
0.16);
  box-shadow: inset 0 0 0 1px #00a2d3, 0 0 0 5px rgba(0, 0, 0, 0.16);
}
.login-button:active:before {
  background: #0591ba;
  background-image: -webkit-linear-gradient(top, #0591ba, #00a2d3);
  background-image: -moz-linear-gradient(top, #0591ba, #00a2d3);
  background-image: -o-linear-gradient(top, #0591ba, #00a2d3);
  background-image: linear-gradient(to bottom, #0591ba, #00a2d3);
}
.login-button:after {
  content: '';
  position: absolute;
  top: 15px;
  left: 12px;
  width: 25px;
  height: 19px;
  background: url("/restaurante/public/images/login/arrow.png") 0 0 no-
repeat;
}

#titulo {
      width: 400px;
      text-align: center;
      margin: 0 auto;
      padding: 40px 0 0 0;
      font-family: 'Helvetica Neue', Helvetica, Arial, sans-serif;
```

```
	font-size: 35px;
	color: #a8a7a8;
}

.erro{
	color: red;
	font-size: 17px;
	text-align: center;
	width: 400px;
	margin: 15px auto;
}

.confirmacao{
	color: blue;
}

#mensagem-login{
	font-size: 20px;
	margin: 20px auto 0px;
}

/* Configuração dos formulários ocultos */

#mask {
 background: url("/restaurante/public/images/login/bg.png");
 width: 100%;
 height: 100%;
 position: fixed!important;
 position: absolute;
 z-index: 9000;
 display: none;
}

.cadastro, .recuperar-senha {
 position: absolute;
 display: none;
 z-index: 9999;
}

.form {
 margin: 0 auto;
 min-width: 300px;
 min-height: 100px;
 display: table;
 background: #fff;
 border: 1px solid #f90;
 padding: 10px;
 margin-bottom: 50px;
}

.form input.button {
 float: right;
}

.form label {
 display: inline-block;
 min-width: 180px;
 text-align: right;
 margin-right: 10px;
}
```

```
.form  p {
 display: block;
 padding: 5px;

}

.form h2 {
 display: block;
 background: #069;
 color: #fff;
 font-size: 24px;
 padding: 5px;
 text-align: center;
 margin-bottom: 10px;
}

.form input.field {
 width: 250px;
 padding: 5px;
 border: 1px solid #069;
 color: black;
 text-shadow: none;
}

.form input {
 margin-left: 5px;
}
```

Pasta “views”,”login”,”js”:

- mostrarForm.js:

```
function openModalForm(classe) {

      $('#mask').fadeIn(500);
      $('#mask').fadeTo("slow",0.5);

      var h = $(document).height();
      var w = $(document).width();

      var fh = h / 2 - $(classe).height() / 2;
      var fw = w / 2 - $(classe).width() / 2;

      $(classe).css('top', fh);
      $(classe).css('left', fw);

      $(classe).fadeIn(1000);

      $('#mask').click(function() {
            $('#mask').hide();
            $(classe).hide();
      });

}
```

- validar.js:

```
$(document).ready(function(){

      function validar_email(email){

            var filtro = /^([\w-]+(?:\.[\w-]+)*)@((?:[\w-]+\.)*\w[\w-
]{0,66})\.([a-z]{2,6}(?:\.[a-z]{2})?)$/i;
            if(filtro.test(email)){
                  return true;
            }
            else {
                  return false;
            }
      }

   $("#form-cadastro").submit(function(){
      var email = $("#mail1").val();
      var senha1 = $("#senha1").val();
      var senha2 = $("#senha2").val();
      var tamanhoSenha = senha1.length;

      var validacao_senha = validar_email(email);

      if(validacao_senha){
        if(tamanhoSenha >= 6){
              if(senha1 == senha2){
                    var url = $(this).attr('action');
                    var data = $(this).serialize();

                    $('#mensagem-cadastro').html('<img
src="/restaurante/public/images/login/ajax-loader.gif">');
                    $.post(url, data, function(o) {
                          if(o == '1'){
                                $('#mensagem-
cadastro').addClass('confirmacao');
                                $('#mensagem-cadastro').html('Cadastro
realizado com sucesso');
                                $("#mail1").val('');
                                $("#senha1").val('');
                                $("#senha2").val('');
                          }
                          else{
                                $('#mensagem-
cadastro').removeClass('confirmacao');
                                $('#mensagem-cadastro').html('Este endereço
de e-mail já está cadastrado');
                          }
                    });
              }
              else{
                    $('#mensagem-cadastro').removeClass('confirmacao');
                    $('#mensagem-cadastro').html('As senhas informadas não
coincidem');
              }
        }
        else{
              $('#mensagem-cadastro').removeClass('confirmacao');
              $('#mensagem-cadastro').html('A senha deve ter no mínimo 6
caracteres');
        }
      }
```

```
      else{
        $('#mensagem-cadastro').removeClass('confirmacao');
        $('#mensagem-cadastro').html('O endereço de e-mail informado é
inválido');
      }
      return false;
   });

   $("#form-esqueci").submit(function(){
         var email = $("#mail2").val();
         var validacao = validar_email(email);

         if(validacao){
               var url = $(this).attr('action');
               var data = $(this).serialize();

               $('#mensagem-recuperar').html('<img
src="/restaurante/public/images/login/ajax-loader.gif">');
               $.post(url, data, function(o){
                        if(o == '1'){
                              $('#mensagem-
recuperar').addClass('confirmacao');
                              $('#mensagem-recuperar').html('Uma nova senha
foi enviada para o seu e-mail');
                              $("#mail2").val('');
                        }
                        else{
                              if(o == '2'){
                                    $('#mensagem-
recuperar').removeClass('confirmacao');
                                    $('#mensagem-recuperar').html('Não foi
possível enviar nova senha para o seu e-mail');
                              }
                              else{
                                    $('#mensagem-
recuperar').removeClass('confirmacao');
                                    $('#mensagem-recuperar').html('O e-mail
informado não está cadastrado no sistema');
                              }
                        }
               });
         }
         else{
               $('#mensagem-recuperar').removeClass('confirmacao');
               $('#mensagem-recuperar').html('O endereço de e-mail
informado é inválido');
        }
        return false;
   });

});
```

Pasta “views”,”user”:

- index.php:

```
<h2>Usuário</h2>

<form method="post" action="<?php echo URL; ?>user/create">
```

```
      <label>Login</label><input type="text" name="login" required><br>
      <label>Senha</label><input type="password" name="password"
required><br>
      <label>Perfil</label>
            <select name="perfil">
                  <option value="caixa">Caixa</option>
                  <option value="cozinha">Cozinha</option>
            </select><br>
      <label></label><input type="submit" value="Enviar" />
</form>

<hr>

<table id="listaDeUsuarios">
<?php
      if(!empty($this->userList)){

            echo '<tr>';
            echo '<td class="bold">Login</td>';
            echo '<td class="bold">Perfil</td>';
            echo '</tr>';

            foreach($this->userList as $key => $value){
                  echo '<tr>';
                  echo '<td>'.$value['login'].'</td>';
                  echo '<td>'.$value['perfil'].'</td>';
                  echo '<td class="link">';
                  echo '<a
href="'.URL.'user/edit/'.$value['user_id'].'">Editar</a> ';
                  echo '<a
href="'.URL.'user/delete/'.$value['user_id'].'">Deletar</a>';
                  echo '</td>';
                  echo '</tr>';
            }
      }

?>
</table>
```

- edit.php:

```
<h1>Usuário: Editar</h1>

<form method="post" action="<?php echo URL; ?>user/editSave/<?php echo
$this->user['user_id'];?>">
      <label>Login</label><input type="text" name="login" value="<?php echo
$this->user['login'];?>" required><br>
      <label>Senha</label><input type="password" name="password"
required><br>
      <label>Perfil</label>
            <select name="perfil">
                  <option value="caixa" <?php if($this->user['perfil'] ==
'caixa') echo 'selected'; ?>>Caixa</option>
                  <option value="cozinha" <?php if($this->user['perfil'] ==
'cozinha') echo 'selected'; ?>>Cozinha</option>
            </select><br>
      <label></label><input type="submit" value="enviar" />
</form>
```

Pasta “views”,”user”,”css”:

- default.css:

```
/** Forms */

#listaDeUsuarios {
	border-collapse: collapse;
}

form label {
	display: block;
	float: left;
	width: 160px;
	padding: 0px;
	margin: 10px 0px 0px;
	text-align: right;
}

form input, form select {
	width: auto;
	margin: 5px 0 0 10px;
	padding: 4px;
}

.link {
	border: none;
}

td {
	border: 1px solid black;
	vertical-align: top;
	padding: 5px;
}

.bold {
	font-weight: bold;
}
```

Arquivo “index.php”:

```
<?php

require_once 'libs/bootstrap.php';
require_once 'libs/controller.php';
require_once 'libs/model.php';
require_once 'libs/view.php';

require_once 'libs/database.php';
require_once 'libs/session.php';
require_once 'config/paths.php';
require_once 'config/database.php';

$app = new Bootstrap();
```

Arquivo “.htaccess”:

```
php_flag display_errors on
php_value error_reporting 9999

RewriteEngine On

RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-l

RewriteRule ^(.+)$ index.php?url=$1 [QSA,L]
```