UNIVERSIDADE FEDERAL DO PIAUÍ
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
CURSO BACHARELADO EM SISTEMAS DE INFORMAÇÃO

DESENVOLVIMENTO DE UM SISTEMA *WEB* PARA CONTROLE E GERENCIAMENTO DE RESTAURANTES COM PRINCÍPIOS DE RESPONSIVIDADE

KAIO CÉSAR MARCOS DE MOURA

PICOS - PI
2016

KAIO CÉSAR MARCOS DE MOURA

DESENVOLVIMENTO DE UM SISTEMA *WEB* PARA CONTROLE E GERENCIAMENTO DE RESTAURANTES COM PRINCÍPIOS DE RESPONSIVIDADE

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros (CSHNB) da Universidade Federal do Piauí (UFPI) como parte dos requisitos para obtenção do Grau de Bacharel em Sistemas de Informação, sob orientação da Professora Ma. Alcilene Dalília de Sousa

PICOS - PI

2016

FICHA CATALOGRÁFICA

Serviço de Processamento Técnico da Universidade Federal do Piauí

Biblioteca José Albano de Macêdo

M929d Moura, Kaio César Marcos de.

Desenvolvimento de um sistema *web* para controle e gerenciamento de restaurantes com princípios de responsividade / Kaio César de Moura.– 2016.

CD-ROM : il.; 4 ¾ pol. (46 f.)

Monografia (Curso Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí, Picos, 2016.

Orientador(A): Prof.ª Ma. Alcilene Dalília de Sousa

DESENVOLVIMENTO DE UM SISTEMA *WEB* PARA CONTROLE E GERENCIAMENTO DE RESTAURANTES COM PRINCÍPIOS DE RESPONSIVIDADE

KAIO CÉSAR MARCOS DE MOURA

Monografia aprovada como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 19 de fevereiro de 2016

Alcilene Dalília de Sousa
Profª. Ma. Alcilene Dalília de Sousa
Orientadora

Prof. Esp. Ivenilton Alexandre de Souza Moura
Membro

Prof. Me. Antonio Oseas de Carvalho Filho
Membro

Dedico este trabalho a minha família, em especial aos meus pais, Maria Marcos de Moura e Francisco das Chagas Costa, por nunca desistirem de mim e por todos os sacrifícios que fizeram para que eu chegasse até aqui,

AGRADECIMENTOS

Agradeço à Deus por me prover os insumos para enfrentar todos os obstáculos que encontrei no decorrer dessa fase.

A minha família, em especial aos meus pais, que nunca permitiram que eu desistisse dos meus objetivos. Pela educação, o amor e o respeito, e acima de tudo, pelo ser humano que me tornei.

A minha orientadora, Alcilene Dalília, que me recebeu de braços abertos e me ajudou com a mais pura vontade. E ao meu primeiro orientador, e mestre para a vida, Ivenilton Alexandre, que me ensinou coisas que levarei para a minha vida, e acima de tudo, por nunca perder a fé em mim, mesmo quando eu mesmo perdia.

A todos os professores do curso de Sistemas de Informação. Que além de me passarem conhecimento, me permitiram ver o mundo por uma outra perspectiva.

Acima de tudo, agradeço aos meus colegas de curso, que sempre estiveram ao meu lado e sempre me ajudaram a continuar lutando. Que apesar das extremas diferenças, me permitiram viver os melhores anos da minha vida. Pelos risos, choros, desesperos, abraços, palavras de conforto, e abalos psicológicos. Para sempre em meu coração, Laronso, Gilberlon, Givanaldo, Viviane, Lívia, Isabel, Celles, Pamela, Abimael (Meu Herói), Janaína e Renan.

“Não há solidão mais triste do que a do homem sem amizades.”

Francis Bacon

“Não há nada que você não possa aprender.”

Ivenilton Alexandre

“É preciso tentar não sucumbir sob o peso de nossas angústias, e continuar a lutar.”

Alvo Dumbledore.

RESUMO

O presente trabalho tem como objetivo o desenvolvimento de um sistema *web* para gerenciamento de restaurantes. O projeto foi desenvolvido através da percepção de problemas na gestão de informações dentro de um restaurante. Uma organização do ramo pode utilizar da ferramenta para ajudar a gerir e melhor controlar o fluxo de informações, podendo assim automatizar algumas atividades. Os conceitos de sistemas de informações são de extrema importância para o desenvolvimento do projeto. Assim como também o aproveitamento dos modelos de engenharia de *software*, *design* responsivo, os princípios de desenvolvimento de sistemas *web* e suas ferramentas. Com este trabalho concluído, conseguiu-se um sistema de gerenciamento capaz de realizar as principais atividades de gestão em um restaurante e ainda dar agilidade e confiabilidade no trato das informações envolvidas nestas atividades.

Palavras-chave: Sistemas de informações, Sistema *Web*, Gerenciamento de restaurantes.

ABSTRACT

This work aims to develop a web system for managing restaurants. The project was developed through the perception of problems in information management in a restaurant. An organization in the industry can use the tool to help better manage and control the flow of information, thus being able to automate some activities. The concepts of information systems are of utmost importance for the development of the project. As well as the use of software engineering models, responsive design, principles of web development systems and tools. With this work completed, it was possible a management system capable of performing the main management activities at a restaurant and still give agility and reliability in handling information involved in these activities.

Keywords*: Information systems, Web System, Ruby On Rails, Control of restaurants.*

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

CERN	*European Council for Nuclear Research*
CSS	*Cascading Style Sheets*
COC	*Convention Over Configuration*
DRY	*Don't Repeat Yourself*
FCS	Fatores Críticos de Sucesso
IE	*Internet Explorer*
IHC	Interação Humano-Computador
IRB	*Interactive Ruby*
MVC	*Model-View-Controller*
RF	Requisitos Funcionais
RN	Regras de Negócio
RNF	Requisitos Não Funcionais
SAE	Sistemas de Automação De Escritórios
SAD	Sistemas de Apoio À Decisão
SGBD	Sistema Gerenciador de Banco de Dados
SI	Sistemas de Informação
SIE	Sistemas de Informação Executiva
SIG	Sistemas de Informação Gerencial
SiGeR	Sistema de Gerenciamento de Restaurantes
SQL	*Structured Query Language*
STC	Sistemas de Trabalho do Conhecimento
STP	Sistema de Processamento de Trabalho
XP	*Extreme Programming*

SUMÁRIO

1 INTRODUÇÃO

As informações estão presentes em todos os ambientes de uma organização, e saber administrá-las é um grande desafio. A criação de um sistema que ajude a gerir todas as informações de forma a organizá-las e usá-las em seu benefício torna-se indispensável nos dias atuais. “Um sistema de informação (SI) é um conjunto de componentes inter-relacionados que coletam, manipulam e disseminam dados e informações para proporcionar um mecanismo de realimentação para atingir um objetivo.” (STAIR; REYNOLDS, 2006).

As organizações passam a utilizar sistemas gerenciadores de informações como uma forma de ganhar agilidade na realizações de processos e tomadas de decisões. Com um bom sistema implantado e com sua correta utilização, essa empresa poderá competir de forma justa no mercado que vem exigindo cada vez mais inovações.

Proporcionar ferramentas e meios para a automação dos processos executados diariamente na organização, através de um sistema *web*, é o principal objetivo desse trabalho. Assim, a administração poderá dispor de informações mais confiáveis e organizadas, podendo ajudar na tomada de decisões e no controle real da sua empresa. Com essa poderosa ferramenta em mãos, e com o seu uso apropriado qualquer restaurante se encontrará ainda mais capacitado e preparado para o mercado, se tornando um competidor à altura dos grandes empreendimentos na área.

Além de proporcionar essas soluções, o trabalho também visa implementar funcionalidades que permitirão a criação de novos serviços na empresa. Como o apoio de um sistema *web* um cadastro de cliente, pedidos de reservas e até um *delivery* (serviço de entrega de pedidos à domicílio) poderão ser oferecidos aos clientes.

1.1 Objetivo

O objetivo desse trabalho é desenvolver um sistema *web* para o gerenciamento de restaurantes, com a finalidade de automatizar e agilizar o controle e o gerenciamento dos recursos da organização, como também para dar suporte no setor de vendas e ajudar na tomada de decisões.

1.2 Organização do Documento

Após uma breve introdução e o esclarecimento do objetivo, este trabalho é apresentado em 5 capítulos. São estes:

- Capítulo 2 – Referencial Teórico: Fornece o embasamento teórico para o trabalho. São mostrados conceitos relacionados a sistemas de informação, engenharia de software e tecnologias utilizados para o desenvolvimento do mesmo;
- Capítulo 3 - Sistema *Web* Para Controle e Gerenciamento e Restaurantes com Princípios de Responsividade: Será mostrada a análise de requisitos, os diagramas produzidos para o entendimento do problema e o desenvolvimento do sistema;
- Capítulo 4 – Resultados Obtidos: será mostrado os resultados alcançados com a aplicação das tecnologias e métodos apresentados nos capítulos anteriores. Através de imagens de telas do sistema, será possível ver como o mesmo funciona;
- Capítulo 5 - Conclusões e Trabalhos Futuros: Apresenta-se a conclusão do trabalho e indicações de trabalhos futuros que possam ser desenvolvidos com o intuito de melhorar a aplicação.

2 REFERENCIAL TEÓRICO

Neste capítulo é apresentado, os conceitos necessários ao desenvolvimento desse trabalho. Uma organização que prioriza o gerenciamento e o controle de suas informações possuem maior facilidade e confiança na tomada de decisões. Um bom controle e uma boa gerência desses dados podem proporcionar uma grande vantagem no mercado.

Um bom sistema de informação ajudaria nessas tarefas de forma a amenizar os riscos e reduzir o trabalho na hora de organizar todas as informações que circulam no ambiente. Com isso evitar possíveis problemas que possam causar prejuízos na administração da empresa.

2.1 Sistemas de Informação

Os sistemas de informação quando bem implantados podem prover inúmeros benefícios para uma organização. Para O'Brien (2006, p.6), "Sistema de informação é um conjunto organizado de pessoas, hardware, software, redes de comunicações e recursos de dados que coleta, transforma e dissemina informações em uma organização.".

As informações processadas e distribuídas em um sistema de informação poderão ser acessadas e utilizadas por todos os níveis da organização, desde que sejam relevantes para este. Esse sistema dará suporte a todos os funcionários, podendo ser de extrema importância na solução de problemas como na tomada de decisões.

Segundo Rezende e Abreu (2003), os atuais sistemas de informação têm alguns pontos em comum, independentemente do negócio da empresa, como lidar com grande volume de informação, processamentos complexos, interligação de diversas tecnologias e auxílio na qualidade, produtividade e competitividade organizacional. O uso desses sistemas nas organizações proporciona uma grande vantagem competitiva no mercado, além de manter um crescimento progressivo.

Porém, o simples fato de implantar um sistema de informação não implicará no sucesso da organização. Um sistema sem a correta utilização e/ou alimentado com informações equivocadas podem prejudicar o trabalho que vem sendo realizado pela empresa e até levá-la a sua ruína. Por isso é de extrema importância o acompanhamento e a constante avaliação do processo de implantação do sistema, para que o mesmo venha a provir insumos para a organização ascender no mercado.

2.1.1 Classificação dos Sistemas

Mesmo dentro de uma organização, é comum encontrar diferentes tipos de sistemas trabalhando de forma interativa. Segundo Mulbert e Ayres (2005), os sistemas podem ser classificados de vários modos, não existindo uma única classificação rígida. É comum essa classificação ser dada por meio de características da organização. Ou seja, as organizações são divididas em níveis de hierarquia, onde cada nível possui um grupo de profissionais com responsabilidades e funções específicas.

De acordo com as autoras, a classificação segundo os níveis organizacionais é a mais abordada pelas literaturas especializadas em sistemas de informação. Nela as categorias específicas de sistemas dão suporte a cada um dos níveis hierárquicos: operacional, gerencial, estratégico e de conhecimento.

- Sistemas de nível operacional

  Nesse nível, os sistemas dão suporte às atividades e transações rotineiras da organização, o chamado chão de fábrica. Eles são denominados de sistemas de processamento de transações (STPs).

  Os STPs são altamente estruturados, pois tanto os dados coletados para entrada quanto as regras de processamento são previamente conhecidas. Esses sistemas podem ser vitais para algumas empresas, pois a interrupção do seu funcionamento pode causar danos irreparáveis à organização. São exemplos: sistemas de cadastro de vendas, sistema de reserva de passagens, folha de pagamento, etc.

- Sistemas de nível gerencial

  Os sistemas de nível gerencial darão suporte às atividades rotineiras, gerenciando e controlando estas. Eles serão alimentados pelos STPs e então serão gerados relatórios que ajudarão os gerentes na tomada de decisões. Os dois tipos principais desses sistemas são os sistemas de informação gerencial (SIGs) e os sistemas de apoio à decisão (SADs).

  Os SIGs fornecerão aos gerentes relatórios padronizados contendo as informações sobre a situação atual da empresa. O enfoque dos relatórios gerados por esses sistemas é a tomada de decisão estrutural, que são previamente

conhecidas. Os variados tipos de relatórios gerados por esses sistemas servirão como uma resposta rápida à consultas realizadas sobre a organização.

Por outro lado, temos os SADs que irão dar suporte aos gerentes na hora de tomar decisões não rotineiras. Ao utilizar um desses sistemas, o usuário está procurando pelas informações que lhe ajudarão a resolver determinados problemas inesperados e complexos. Um SAD possui três elementos principais, gerenciamento de dados, de modelos, e a interface do usuário.

- Sistemas de nível estratégico

  Os sistemas nesse nível dão suporte à tomada de decisões na alta administração, ou seja, aos diretores, proprietários, sócios, etc. Segundo Rezende (2005) esses sistemas trabalham com os dados no nível macro, filtrados das operações das funções organizacionais da organização, considerando ainda o meio ambiente interno ou externo.

  O principal exemplo de sistema nesse nível são os sistemas de informação executiva (SIE). Eles disponibilizam acesso às informações de forma rápida e atualizada, muitas vezes utilizando de recursos visualmente amigáveis, gráficos e também apresentações multimídia.

- Sistemas de nível de conhecimento

  Nesse nível, os sistemas buscam auxiliar a empresa na integração de novas tecnologias e a controlar o fluxo de documentos. Os sistemas de trabalho do conhecimento (STC) auxiliam no processo de criação de informações. Outro exemplo, são os sistemas de automação de escritórios (SAE) que irão melhorar a comunicação, a produtividade e organização de documentos no trabalho de equipes dentro da organização.

2.1.2 Sistemas de Informação Gerencial

De acordo com Stair e Reynolds (2006, p.21) um SIG é um conjunto organizado de pessoas, procedimentos, *software*, bases de dados e dispositivos, usados para fornecer informações rotineiras a gerentes e tomadores de decisões. Os SIGs tem o foco basicamente na

eficiência operacional. Eles recebem entradas de dados provenientes dos sistemas de processamento de transações.

As saídas geradas pelos SIGs são em formatos de relatórios que determinam a exata situação atual da organização. Esses relatórios dão suporte à tomada de decisão gerenciais dos níveis mais altos de gerenciamento, onde as decisões tendem a ser semiestruturadas e menos rotineiras. Os principais tipos de relatórios gerados por esses sistemas são:

- Relatórios programados: como o nome diz, são aqueles relatórios que são produzidos periodicamente ou de forma planejada. Podem ser diários, semanais ou mensais.
- Relatório indicador de pontos críticos: resume-se às atividades críticas do dia anterior e fica disponível caracteristicamente a cada dia de trabalho. Estão ligados geralmente aos fatores críticos de sucesso (FCS) de uma organização.
- Relatórios sob solicitação: são gerados para dar certas informações a pedido de um administrador. São relatórios específicos para atender a determinada solicitação do gerente.
- Relatórios de exceção: são os relatórios produzidos automaticamente quando uma situação é incomum ou requer uma atitude da administração. Eles são frequentemente usados para monitorar aspectos críticos para o sucesso de uma organização.

Diante do contexto, um SIG é de extrema importância para uma organização. Ele coleta e gerencia todas as informações importantes e gera relatórios que ajudarão na tomada de decisões de seus gerentes. Uma empresa atualmente sem um sistema desses fica à mercê da inexatidão de suas informações. Um SIG bem utilizado pode alavancar o sucesso de uma organização, dando aos seus administradores controle total de sua empresa.

Um sistema pode ser dividido em sistema *desktop* (*Offline*) ou sistema *web* (*Online*). Isso vai depender das especificações que o cliente deseja e em que ambiente esse sistema irá ser executado. Enquanto que os sistemas desktop podem ser executados em praticamente qualquer máquina, os sistemas *web* necessitam de uma plataforma de rede para funcionar.

2.1.3 Sistemas *Web*

O uso de sistemas *web* tem sido cada vez mais comum em diversos contextos. Um sistema de informação *web* nada mais é do que um sistema convencional que necessita da

estrutura de rede para funcionar. Cabe ao desenvolvedor, através do levantamento de requisitos e das necessidades do cliente, decidir qual tipo de sistema se encaixará melhor no seu projeto.

"O dinamismo e a criatividade de muitos sistemas *web* torna a qualidade no desenvolvimento desses sistemas um aspecto crucial. Para entender, controlar e melhorar a qualidade de sistemas *web* é necessário utilizar métodos modelos e técnicas de engenharia adequadas." (OLSINA *et al*., 2001).

Uma das grandes vantagens dos sistemas *web* é a portabilidade. Você pode acessar o seu sistema de qualquer lugar do mundo desde que tenha um ponto de acesso de internet. Assim, passa a economizar com infraestrutura e compra de equipamentos.

Afim de buscar aumentar essa portabilidade a indústria encontrou uma forma de "colocar" os computadores dentro do seu bolso. A inclusão dos dispositivos móveis fez com que as pessoas pudessem desfrutar das vantagens da *internet* em qualquer lugar. Porém era clara a necessidade da adaptabilidade da *web* para esses novos formatos e tamanhos de telas, assim não seria necessário abrir mão do conteúdo afim de ter a comodidade. (LOPES, 2013).

2.1.4 *Web Design* Responsivo

Para Zemel (2012, p. 17), o *web design* responsivo é aquele *web design* que "responde" a quaisquer dispositivos/resoluções e, devido a uma série de características técnicas bem específicas, é bem apresentado em qualquer um deles. Dessa forma é possível visualizar um página/aplicação *web* a partir de qualquer dispositivo conectado, sem perder conteúdo ou a necessidade de maior esforço para visualizá-lo.

Zemel (2012) ainda mostra que existem três tecnologias principais para o desenvolvimento de um *web design* responsivo. São elas:

- *Layout* fluído: parte do preceito da não especificação de medidas fixas para o *layout* do projeto. Assim a adaptação do que é apresentado na tela torna-se algo automático. As duas medidas flexíveis mais usadas são as porcentagens e os *em*, unidade escalável*;*
- Imagens e recursos flexíveis: independente do dispositivo utilizado é de extrema importância que os *assets* (recursos como imagens, vídeos, etc) também se adaptem as diferentes resoluções. Assim, através de técnicas variadas, é possível garantir uma melhor experiência na utilização da aplicação;

- *Media Queries*: permite, através do CSS, que determinadados elementos possam ser reposicionados, retirados ou colocados, dependendo da resolução da tela do dispositivo. Com isso é possível observar grandes diferenças na aparência da mesma aplicação em diferentes dispositivos.

Com a utilização correta destas tecnologias juntamente com as linguagens de marcação e estilização HTML, CSS e Java *Script* é possível desenvolver um bom *design* responsivo. De acordo com o trabalho, como o mesmo será aplicado e quais tecnologias serão necessárias para desenvolvê-lo, a utilização da responsividade pode ser o diferencial que destacará dos demais no mercado.

2.2 Tecnologias de Desenvolvimento

Diversas tecnologias e métodos são utilizadas no desenvolvimento de uma aplicação, como: a linguagem de programação, *Ruby*; o *framework* para desenvolvimento *web*, *Ruby on rails;* o *framework* para *front-end*, *twitter bootstrap*; entre outros. Cada tecnologia possui tarefas específicas e juntas colaboraram para os resultados do trabalho.

2.2.1 Engenharia de Software

Atualmente os softwares estão sendo utilizados em praticamente todas as áreas de trabalho. Com esse avanço faz-se necessário a criação de métodos e processos para a criação de softwares mais eficientes e de qualidade. Segundo Pressman (2011, p.24) o processo de software pode ser definido como uma coleção de padrões que definem um conjunto de atividades, ações, tarefas de trabalho, produtos de trabalho e/ou comportamentos relacionados necessários ao desenvolvimento de softwares de computador.

"A engenharia de software é uma disciplina da engenharia que se ocupa de todos os aspectos da produção de software, desde os estágios iniciais de especificação do sistema até a manutenção desse sistema, depois que ele entrou em operação." (SOMMERVILE, 2011, p.5). Para se conseguir uma engenharia de software com alta qualidade faz-se necessário a utilização de modelos.

Pressman (2011) afirma que os modelos prescritivos serão utilizados para dar ordem no processo de desenvolvimento do softwares. Eles prescrevem um conjunto de atividades e um fluxo para utiliza-las. Os principais modelos prescritivos são:

- Cascata: sugere uma abordagem sistemática e sequencial que começa com a especificação dos requisitos e progride ao longo do planejamento, modelagem, construção e implementação. Sugerido apenas quando os requisitos são bem entendidos e as mudanças durante o processo de projeto serão limitadas.
- Incremental: o software é produzido com uma série de versões de incrementos. Essas versões possuem prazos, geralmente apertados, para serem entregues.
- Espiral: o processo é representado como uma espiral ao invés de uma sequência de atividades com retornos. Cada loop na espiral representa uma fase do processo. Os riscos são avaliados explicitamente e resolvidos no decorrer do processo.
- Baseado em reuso: demanda uma abordagem iterativa, porém compõe aplicações a partir de componentes de software previamente preparados.

O grande desafio dos modelos prescritivos citados acima é a grande quantidade de tempo utilizada no planejamento e documentação dos projetos. Então como solução surgiram os modelos ágeis de desenvolvimento, que reduzem a burocracia e focam nas mudanças que podem aparecer no decorrer do projeto. "Os modelos ágeis destinam-se a entregar o software rapidamente aos clientes, em funcionamento, e estes podem, propor alterações e novos requisitos a serem incluídos nas iterações posteriores do sistema." (SOMMERVILLE, 2011, p.40).

Alguns exemplos de métodos ágeis de desenvolvimento:

- XP (*Extreme Programming*): "Inclui um conjunto de regras e práticas que ocorrem no contexto de quatro atividades de arcabouço: planejamento, projeto, codificação e teste." (PRESSMAN, 2011). É uma disciplina de desenvolvimento baseada em valores de simplicidade, comunicação, feedback e coragem.
- *Scrum*: é um método ágil genérico mas seu foco é na gerência de desenvolvimento iterativo ao invés de práticas ágeis específicas. O produto é dividido, então cada equipe realiza determinada parte. Assim a comunicação e a flexibilidade do projeto são garantidas.
- Desenvolvimento de software adaptativo: se concentra na colaboração humana e na auto organização das equipes. São definidos planos de ciclos adaptáveis e à medida que são executados os planos podem sofrer ajustes para a realidade da equipe de desenvolvimento (PRESSMAN, 2011).

A aplicação de algum desses modelos acima na produção de um software além de organizar o projeto, permite que a sua produção ocorra de forma ágil e controlada. A engenharia de software tornasse cada vez mais essencial no desenvolvimento de sistemas que buscam a satisfação do cliente como também um auto grau de qualidade.

2.2.2 *Scrum*

Jeff Sutherland e Ken Schwaber criaram o *scrum* há vinte anos com o objetivo de ser uma forma mais rápida, eficaz e confiável de criar softwares para o setor de tecnologia. A escolha do método de desenvolvimento se deu através dos ótimos resultados durante o desenvolvimento de trabalhos passados, e pela forma eficiente de trabalho do mesmo.

De acordo com Sabbagh (2013) o *scrum* é um *framework* ágil que caracteriza-se pela sua clareza, simplicidade e bom desempenho em diferentes contextos, quando utilizado de forma conjunta com outras técnicas e práticas. O método lista uma série de benefícios agregados com sua correta utilização, entre eles estão:

- Entregas frequentes de retorno ao investimento dos clientes.
- Redução dos riscos do projeto.
- Maior qualidade no produto gerado.
- Mudanças utilizadas como vantagem competitiva.
- Visibilidade do progresso do projeto.
- Redução do desperdício.
- Aumento de produtividade.

Para Pressman (2011) o scrum possui um conjunto de padrões de processos onde cada um desses padrões possui um conjunto de ações de desenvolvimento. Os padrões de processos são:

- *Backlog*: Lista de requisitos ou funcionalidades que estão dispostos em ordem de prioridade para serem implementados. Essa lista pode ser atualizada a qualquer momento;
- *Sprints*: São unidades de trabalhos que implementam um requisito do *backlog* e possui um prazo para ser entregue;
- Reuniões *Scrum*: Reuniões pequenas realizadas todos os dias. É realizada para se saber como anda o desenvolvimento ou se há algum problema;

- *Demos*: Entrega de parte do software ao cliente para que a funcionalidade implementada possa ser utilizada e avaliada por ele. A *demo* não precisa necessariamente ter toda a funcionalidade planejada, mas pode ter funções que podem ser entregues respeitando o prazo combinado.

Uma boa metodologia de desenvolvimento bem aplicada pode ser o diferencial para a realização de um projeto. Porém, o mesmo seria inútil se não fosse utilizado com ferramentas e tecnologias de qualidade. Daí entra a responsabilidade do desenvolvedor de escolher uma linguagem de programação adequada, um bom banco de dados, *frameworks* adequados, entre outros.

2.2.3 Linguagem *Ruby*

A linguagem de programação *web Ruby* tem sido bastante utilizada pelos desenvolvedores nos últimos anos. Ela foi criada por Yukihiro Matsumoto em 1995, com o objetivo de ser uma linguagem mais legível e agradável de se programar. Segundo Menegotto e Mierlo (2002) o *Ruby* possui algumas características peculiares, entre elas podemos citar:

- É linguagem interpretada: não precisa ser recompilada sempre que executar os programas novamente, basta executá-lo através do interpretador;
- É puramente orientada a objetos: tudo em *Ruby* é um objeto, inclusive as classes. Mas também tem suporte à programação funcional, imperativa e reflexiva.
- Sintaxe simples: fácil de ler e ser entendida, ajuda no desenvolvimento e manutenção de sistemas escritos com ela.
- Altamente portável: multiplataforma, isto significa que um programa *Ruby* executa sem nenhuma mudança em qualquer plataforma.
- Fortemente tipada e dinâmica: não é necessário definir os tipos de todas variáveis para o sucesso das operações. Também permite que o tipo da variável possa ser alterado durante a execução do programa.

O *Ruby* possui um gerenciador de pacotes e dependências muito eficiente chamado *Ruby Gems*. Os *gems* podem ser vistos como bibliotecas reutilizáveis de código *Ruby* que podem conter também códigos C, Java, .Net.

Outra ferramenta muito importante para a linguagem *Ruby* é o IRB (*Interactive Ruby*). Ele é um console que avalia cada linha inserida e mostra o resultado imediatamente. Assim, o programador poderá realizar vários testes em seu código em tempo real.

Segundo Souza (2012), o *Ruby* se popularizou entre os programadores destacando-se principalmente pela performance, simplicidade e elegância. Com qualidades como essas, e com uma boa performance para aplicações *web* anteriormente comprovada, a escolha dessa linguagem torna-se clara. Atualmente encontra-se entre as linguagens mais usadas, mas muito também se deve à propagação do seu principal *framework,* o *Ruby on Rails.*

2.2.4 *Framework Ruby on Rails*

Segundo Fuentes (2012), um *framework* é um conjunto de *scripts* (instruções a serem seguidas) que dão suporte a uma programação mais rápida, simples e eficiente, possuindo funções, rotinas e variáveis que poupam bastante tempo de desenvolvimento.

O *Ruby on Rails* foi criado por David Heinemeier Hansson e oficialmente apresentado em 2004. O *framework*, escrito na linguagem *Ruby*, foi extraído de um produto de sua empresa, o *Basecamp*. Ele tem como principal objetivo tornar a programação *web* mais fácil, permitindo a realização de várias tarefas com menos código possível.

O *Ruby on Rails* adota uma arquitetura *Model-View-Controller* (MVC). "MVC é um padrão arquitetural onde os limites entre seus modelos, suas lógicas e suas visualizações são bem definidos, sendo muito mais simples fazer um reparo, uma mudança ou uma manutenção, já que essas três partes se comunicam de maneira bem desacoplada." (CAELUM, 2014).

- Modelos (*Model*): auxiliam na busca e no armazenamento no banco de dados, além de compor as regras de negócio, como cálculos e outros procedimentos.
- Apresentação (*View*): é como o aplicativo mostra o resultado das operações e os dados. Normalmente representada em uma página *web*.
- Controle (*Controller*): adquire os dados que vem de parâmetros na URL e/ou de um formulário e repassa para os modelos, que vão fazer o trabalho pesado. Em seguida, adquire o resultado e transforma da maneira adequada para a apresentação.

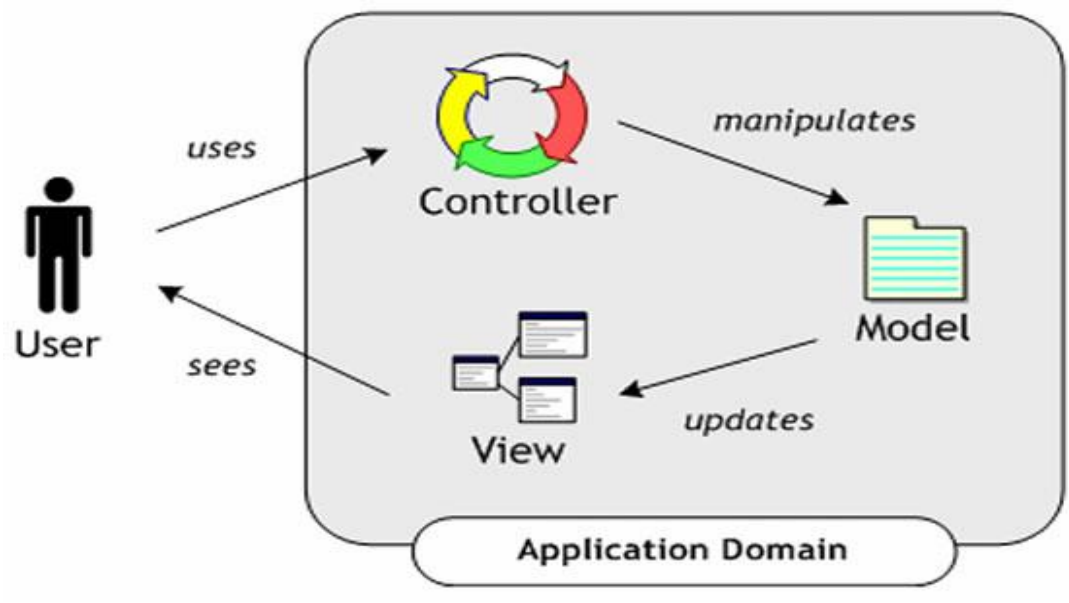


Figura 2 – Arquitetura MVC
Fonte: < http://www.escolacriatividade.com/php-orientado-a-objetos-mvc-em-php/>

A filosofia do *Rails* adota alguns princípios básicos, dentre eles podemos destacar o *Don't Repeat Yourself* (DRY) e o *Convention over Configuration* (COC). O primeiro afirma que nunca se deve fazer mais de uma vez o que não for necessário. Já o segundo diz que ao invés de criar vários arquivos de configuração, apenas se adota uma convenção a ser seguida.

2.2.5 *Framework* de *Front-end Bootstrap*

Bootstrap é um *framework* de código aberto desenvolvido por Mark Otto e Jacob Thornton que a lançaram enquanto eram funcionários do *Twitter*, pois havia uma necessidade de padronizar os conjuntos de ferramentas de *front-end* de engenheiros em toda a empresa (SPURLOCK, 2013).

Ao utilizar o *bootstrap* o desenvolvedor não precisa definir seus elementos básico de interface do zero. Além de ser compatível com HTML5 e CSS3 as suas mais novas versões permitem a criação de *layouts* responsivos. Como toda ferramenta, também possui suas vantagens e desvantagens. Conhecendo estas, cabe ao desenvolvedor decidir se a utilização desse *framework* será necessária para a sua aplicação.

- Vantagens:
  - Possui documentação detalhada e de fácil entendimento;
  - É otimizado para o desenvolvimento de *layouts* responsivos;

    - Possui componentes suficientes para o desenvolvimento de qualquer site ou sistema *web* com interface simples;
    - Facilita a criação e edição de *layouts* por manter padrões;
    - Funciona na maioria dos navegadores atuais (*Chrome*, *Safari*, *Firefox*, IE, *Opera*).
- Desvantagens:
    - Seu código terá de seguir os "padrões de desenvolvimento *Bootstrap*";
    - Tema padrão e comum do *Bootstrap* (caso não faça ajustes visuais, seu projeto se parecerá com outros que também utilizam o *Bootstrap*).

Como apresentado este *framework* é altamente recomendado para todo tipo de desenvolvedor, seja este iniciante que possui conhecimentos básicos em HTML, até o avançado que busca por agilidade e eficiência. O *Bootstrap* possui ferramentas e recursos suficientes para desenvolver um *site* e/ou um sistema *web* por completo.

2.2.6 *Hyper Text Markup Languange* (HTML)

O HTML é uma linguagem de marcação utilizada para o desenvolvimento de páginas web. Criada em 1991, por Tim Berners-Lee, no European Council for Nuclear Research (CERN) na Suíça, foi projetado para realizar a comunicação entre instituições de pesquisa próximas, e compartilhar documentos com facilidade. Somente em 1992 foi liberada para o uso mundial, junto com a biblioteca de desenvolvimento WWW (World Wide Web).

Atualmente o HTML encontra-se na sua versão 5 e de acordo com Silva (2014), proporciona, entre outras, as seguinte vantagens no desenvolvimento *web*:

- Código claramente mais semântico, estando agora mais simples e fácil de ser compreendido;
- Melhor estruturação do código, de forma a facilitar a leitura e o suporte;
- Melhor acessibilidade, sendo capaz de ser executado em dispositivos de baixa potência;
- Melhores benefícios para estratégias de otimização de *sites*.

A linguagem consiste basicamente de um conjunto de *tags* que irão definir como o texto será mostrado no navegador. Junto com o CSS, o HTML formam a base para o desenvolvimento *front-end*, ou seja, as páginas de apresentação de uma aplicação.

2.2.7 *Cascading Style Sheets* (CSS)

Devido à grande popularização do HTML, o mesmo começou a ficar sobrecarregado e complexo. Então em 1996, Hakon Wium Lie, lança a primeira versão do CSS, que visava otimizar a estilização de páginas web. Ele separa os estilos das marcações do texto, facilitando a manutenção das páginas, além de permitir o controle de layouts de vários documentos a partir de um simples documento.

O CSS possui sua própria sintaxe, com algumas de suas propriedades parecidas às do HTML. Na sua forma mais simples, é composto por 3 partes:

- Seletor: *tag* em HTML que terá tal valor aplicado àquela propriedade citada. Cada seletor possui mais de uma propriedade que pode vir a ser modificada e pode ser aplicado a uma classe;
- Propriedade: atributo que será modificado ao receber tal valor;
- Valor: característica que a propriedade irá assumir. Quando os valores forem numéricos, há a opção de determiná-los em pixels, centímetros ou porcentagens.

Atualmente a CSS encontra-se na sua versão 3 (CSS3) agregando diferentes módulos e funcionalidades de estilização que vieram para facilitar as obrigações do desenvolvedor. Entre as principais novas funcionalidades encontram-se: *text-shadow*, *box-shadow*, *linear-gradient*, *opacity*, *border-image*, *animation*, e entre essas e outras tem-se também as regras de *media queries*, que possibilitam a criação de folhas de estilo para diversos dispositivos, permitindo a criação de *designs* responsivos (SILVA, 2014).

2.2.8 *JavaScript*

O *JavaScript* foi originalmente desenvolvido por Brendan Eich quando trabalhou na *Netscape* sob o nome de *Mocha*, posteriormente teve seu nome mudado para *LiveScript* e por fim *JavaScript*. É uma linguagem de programação interpretada, com orientação a objetos, e projetada como linguagem de *script* capaz de criar programas e realizar ações em uma página *web* e complementar as capacidades do HTML (criar efeitos, botões animados, sons, etc.).

Com *JavaScript*, podemos criar efeitos especiais nas páginas e definir interatividades com o usuário. É uma linguagem interpretada, e de fácil aprendizagem. Graças a sua compatibilidade com a maioria dos navegadores modernos, é a linguagem de programação mais utilizada (RODRIGUES, 2010).

Segundo Balduino (2013) a escolha do nome *JavaScript* foi uma estratégia inteligente para quebrar a resistência das pessoas em aprender uma nova linguagem e fazer com que a mesma pudesse pegar carona com o sucesso do *java*. Porém, alguns problemas tornaram a linguagem bastante ignorada. Mas com o passar dos anos tornou-se melhor compreendida e aceita, espacialmente devido ao desenvolvimento e popularização de ferramentas como o *node.js*, que permite o desenvolvimento de servidores com *JavaScript*; o *Rhino*, que permite a execução de código *JavaScript*; e o *jQuery*, que simplifica e abstrai o desenvolvimento no *browser*.

2.2.9 *Postgre*SQL

O *Postgre*SQL é um poderoso sistema gerenciador de banco de dados objeto-relacional de código aberto. Desenvolvido pelo Departamento de Ciências da Computação da Universidade da Califórnia em Berkeley, o projeto *POSTGRES* foi liderado pelo Professor Michael Stonebraker, e teve a sua implementação em 1986. Em 1994, Andrew Yu e Jolly Chen adicionaram um interpretador da linguagem SQL ao POSTGRES. E por fim, em 1996 recebe o seu nome atual, *Postgre*SQL, como uma forma de refletir o relacionamento entre o POSTGRES original e as versões mais recentes com capacidade SQL.

A compatibilidade e eficiência mostradas pelo *Postgre*SQL foram importantes para a escolha do SGBD. Devido a problemas encontrados com outros gerenciadores de banco de dados, o *Postgre*SQL mostrou-se perfeitamente capaz de realizar todas as tarefas necessárias sem apresentar nenhuma dificuldade.

Segundo a Documentação do *Postgre*SQL 8.0.0 (2005), o *Postgre*SQL utiliza o modelo cliente-servidor. Uma sessão do *Postgre*SQL consiste nos seguintes processos (programas) cooperando entre si:

- Um processo servidor, que gerencia os arquivos de banco de dados, aceita conexões dos aplicativos cliente com o banco de dados, e executa ações no banco de dados em nome dos clientes. O programa servidor de banco de dados se chama *postmaster*;

- O aplicativo cliente do usuário (*front-end*) que deseja executar operações de banco de dados. Os aplicativos cliente podem ter naturezas muito diversas: o cliente pode ser uma ferramenta no modo caractere, um aplicativo gráfico, um servidor *Web* que acessa o

> banco de dados para mostrar páginas *Web*, ou uma ferramenta especializada para manutenção do banco de dados. Alguns aplicativos cliente são fornecidos na distribuição do *Postgre*SQL, sendo a maioria desenvolvido pelos usuários.

Como um banco de dados de nível corporativo, o PostgreSQL possui funcionalidades sofisticadas como o controle de concorrência multiversionado, recuperação em um ponto no tempo, *tablespaces*, replicação assíncrona, transações agrupadas, cópias de segurança, um sofisticado planejador de consultas e registrador de transações sequencial para tolerância a falhas. Suporta conjuntos de caracteres internacionais, codificação de caracteres *multibyte*, Unicode e sua ordenação por localização, sensibilidade a caixa (maiúsculas e minúsculas) e formatação. É altamente escalável, tanto na quantidade enorme de dados que pode gerenciar, quanto no número de usuários concorrentes que pode acomodar.

Além de ser um banco de dados significativamente grande, é compatível com sistemas operacionais *Windows*, *Linux* e outros. Possibilitando assim a sua utilização em projetos desenvolvidos para várias plataformas.

Com todas tecnologias aqui mencionadas é possível desenvolver um projeto de grande porte, que possa atender as necessidades e resolver problemas dos usuários. Com a união dessas tecnologias e com a ideia aqui apresentada, faz-se possível a realização deste trabalho. Que busca a melhor forma de aplicar tais ferramentas com o intuito de facilitar o trabalho e buscar novos meios de realizar tarefas, mais modernas e eficientes, nesse nicho de mercado.

3 DESENVOLVIMENTO DO SISTEMA

Neste capítulo serão apresentados os requisitos funcionais e não funcionais, e alguns diagramas que demonstrarão como o sistema funciona.

O sistema de gerenciamento de restaurantes, SiGeR, atende duas necessidades básicas do seguimento de restaurantes. O gerenciamento das atividades do estabelecimento e o departamento de vendas, que possui ligação direta com o consumidor.

Utilizando o Ruby como linguagem de programação, e o *framework Ruby on Rails*, foi possível um melhor aproveitamento dos recursos disponibilizados, fazendo com que a aplicação em questão tenha sido desenvolvida com maior facilidade. Com as linguagens de marcação e estilização HTML, CSS e *Java Script* foi possível dar uma melhor apresentação para as páginas, fazendo com que a interação humano-computador (IHC) seja mais apresentável e convidativa. O *framework Bootstrapt* trouxe ferramentas que possibilitaram o melhor desenvolvimento de interfaces como também permitiu o desenvolvimento de um *design web* responsivo. O SGBD *PostgreSQL* trouxe facilidade no trato com os dados armazenados na aplicação. Como possui compatibilidade com diversas plataformas, permite que o sistema possa ser executado na maioria das máquinas e dispositivos. Com a colaboração de todas essas tecnologias, o SiGeR poderá alcançar uma variedade grande de público, permitindo que a organização que o adote possa usufruir de uma ferramenta que permitirá melhorar o seu desempenho em todos os setores.

3.1 Requisitos do Sistema

Através da experiência pessoal no ramo de restaurantes foi-se possível realizar o levantamento de requisitos. Estes se tratam da série de necessidades do cliente para com o *software*, para que o mesmo possa ser construído de acordo com as restrições estabelecidas. O SiGeR trata especificamente de agir como a solução para os principais problemas percebidos nesta fase do desenvolvimento. Os requisitos funcionais, não funcionais e as regras de negócio serão mostrados a seguir, apresentados com um identificador, a descrição que o narra de forma explicativa e as dependências do mesmo.

3.1.1 Requisitos Funcionais

Os Requisitos funcionais (RF) mostram a forma como o sistema deve funcionar e quais atividades o cliente deseja realizar. Esses requisitos referentes ao SiGeR estão dispostos no Quadro 1.

Quadro 1 - Requisitos Funcionais

Identificador	Descrição	Depende de
RF01	O sistema deve realizar o cadastro de usuários atribuindo-os uma função.	
RF02	Cada função de usuário terá níveis de acesso aos recursos diferenciado.	RF01
RF03	O Sistema deverá permitir a solicitação de cadastro de novos clientes.	RF01
RF04	O sistema permitirá o gerenciamento das informações cadastradas.	
RF05	O sistema deverá mostrar o cardápio para o dia da semana atual.	RF04
RF06	O sistema deverá cadastrar os fornecedores que atendem ao estabelecimento, como também os produtos fornecidos por eles.	RF04
RF07	O cliente poderá montar o seu prato de acordo com a sua vontade e disponibilidade do alimento.	RF04
RF08	O cliente poderá adicionar produtos ao seu pedido como também a quantidade de pratos que desejar.	RF06 e RF07
RF09	O sistema deve atualizar o seu estoque de acordo com a saída de produtos e alimentos.	

3.1.2 Requisitos Não Funcionais

Os requisitos não funcionais de software é aquele que descreve não o que o sistema fará, mas como ele fará. O Quadro 2 mostra detalhadamente os requisitos não funcionais do SiGeR.

Quadro 2 – Requisitos Não Funcionais

Identificador	Descrição	Depende de
RNF01	O sistema deve controlar o acesso às funcionalidades. O gerenciamento dos recursos do sistema devem ser acessíveis somente por usuários autenticados.	

RNF02	O sistema deve presar pela segurança das informações cadastradas, evitando que pessoas não autorizadas possam visualizá-las e/ou alteá-las.	RNF01
RNF03	O sistema deve possuir interface intuitiva que permita aos usuários utilizarem a plataforma e realizarem suas atividades sem esforços.	
RNF04	O sistema deve estar disponível pela Internet, e capaz de ser acessado por meio dos principais navegadores disponíveis no mercado.	
RNF05	A persistência das informações deve ser implementada em um Sistema Gerenciador de Bancos de Dados Relacionais (SGBDR) livre.	
RNF06	O sistema deve manter suas informações atualizadas para que não ocorra imprevistos que possam dificultar a utilização do mesmo.	RNF04

3.1.3 Regras de Negócio

As regras de negócio são identificadas pelos requisitos não funcionais que são as normas e condições estabelecidas pelo cliente e que devem ser seguidas na execução de alguma atividade do sistema (GUEDES, 2011). As regras de negócio do SiGeR são apresentadas no Quadro 3.

Quadro 3 – Regras de Negócio

Identificador	**Descrição**	**Depende de**
RN01	Não é permitido a realização de pedidos por usuários não cadastrados.	
RN02	Clientes e visitantes não podem realizar atividades de gestão.	
RN03	Um usuário apenas terá acesso as funções de clientes após a aprovação do seu cadastro pelo administrador.	RN01
RN04	O sistema permite que o cliente possa acrescentar quantos pratos e produtos o cliente desejar, desde que estejam em estoque.	
RN05	Um produto só pode ser cadastrado se o mesmo for vinculado a um fornecedor previamente cadastrado.	

3.2 Principais Diagramas do Sistema

Após a especificação de requisitos é possível realizar a diagramação do sistemas. Aqui o cliente poderá ver, através dos diagramas, o que será desenvolvido de acordo com suas necessidades. Nesse tópico serão apresentados os principais diagramas do SiGeR.

3.2.1 Diagrama de Casos de Uso

Por meio dos diagramas de casos de uso é possível compreender os requisitos e facilitar a compreensão do comportamento do sistema, apresentando uma visão geral das funcionalidades, identificando os usuários e quais seus papéis e funções na interação com a aplicação (GUEDES, 2011).

O SiGeR possui dois tipos de usuários: O Administrador do sistema e os Clientes. O administrador possui acesso total ao sistema, e pode gerir todas as informações além de cadastrar novos usuários. Os Clientes, após serem cadastrados, podem realizar pedidos e acessar o cardápio do dia. O Quadro 4 apresenta os atores da aplicação e uma breve descrição do que eles podem realizar no sistema.

Quadro 4 – Atores do Sistema

Ator	**Descrição**
Usuário não autenticado	Este ainda não pode ter acesso ao sistema, podendo apenas solicitar o seu cadastro para, possivelmente receber um *login*.
Cliente	Após ter o seu cadastro realizado, agora o usuário poderá acessar o sistema. Aqui ele pode consultar o cardápio para ver as opções de alimentos. Se preferir, pode realizar um pedido, criando pratos e adicionando produtos.
Administrador	Possui acesso total as atividades da aplicação. Pode aceitar solicitações de futuros clientes ou cadastrá-los. Pode cadastrar e gerir cardápios, produtos, alimentos, fornecedores.

O diagrama de casos de uso mostrado na Figura 2 apresenta a visão geral do sistema, mostrando cada ator e suas ações no sistema.

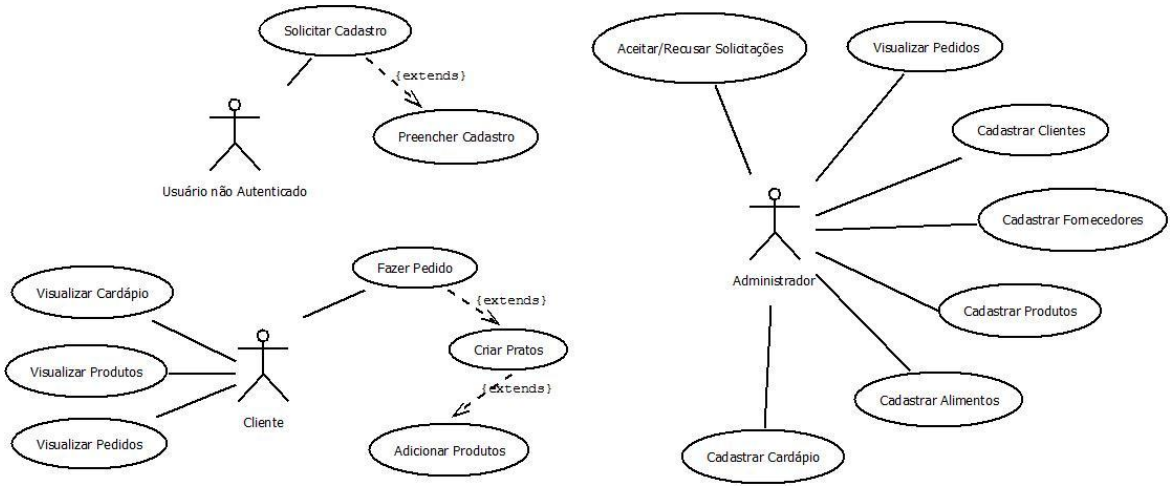


Figura 2 - Diagrama de Casos de Uso da Visão Geral do Sistema

Apresentada a visão geral do sistema por meio dos diagramas de casos de uso, foi possível perceber os atores que utilizarão o *software* e as funcionalidades que estarão disponibilizadas para os mesmos. Com base nas informações disponíveis nos diagramas de caso de uso, foi possível iniciar a construção do diagrama de classes.

3.2.2 Diagrama de Classes

Por meio do diagrama de classes é apresentada a estrutura de classes do sistema e como elas se relacionam entre si, além de definidos os métodos e atributos de cada uma. Dessa forma, tem-se uma referência conceitual sobre as informações necessárias para o desenvolvimento da aplicação.

A Figura 3 apresenta o diagrama de classes do SiGeR, bem como suas classes, atributos e métodos e ainda como as mesmas se associam e trocam informações. As classes são representadas pelos retângulos que carregam seu nome na parte superior, seus atributos na parte central e seus métodos na parte inferior.

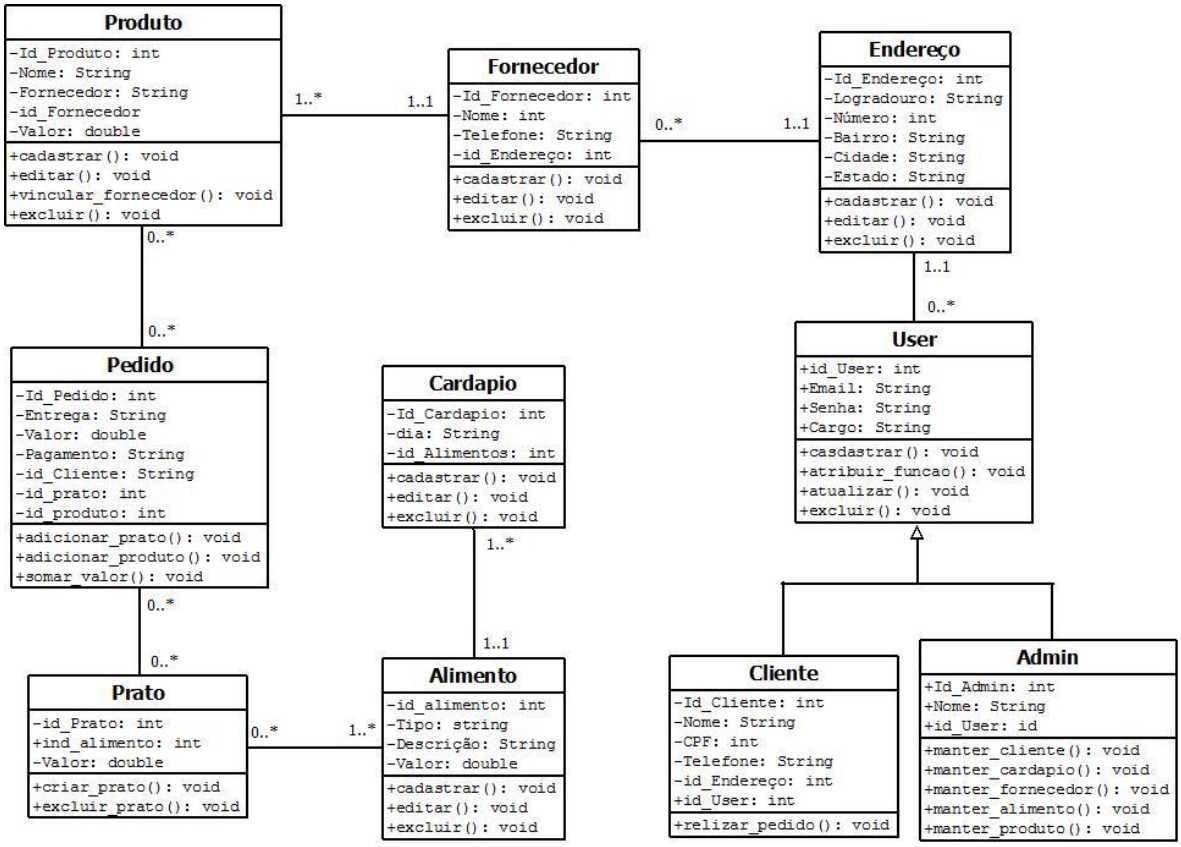


Figura 3 - Diagrama de Classes

A classe “User” manterá as informações dos usuários. Ela se relaciona com a classe “Endereco” que por sua vez mantém as informações de endereço deste. User também possui uma relação especial de generalização/especialização com as classes “Cliente” e “Administrador” que herdam suas características. O método “atribuir_função” é o responsável por determinar qual o nível de acesso daquele usuário. Cliente possui o método “realizar_pedido” que será através deste que ele realizará sua principal atividade. O Administrados possui todos os método de gestão relacionados às demais tabelas.

As tabelas “Produtos” e “Fornecedor” estão relacionadas através do método “vincular_fornecedor”, que dirá a qual fornecedor cada produto está ligado. A tabela “Alimento” está ligada a “Cardápio” e “Prato”, onde no primeiro permite vincular aquele alimento a um determinado dia da semana para que aparece do cardápio correspondente; e no segundo onde, juntamente com outros alimentos, permite a criação de um prato.

Por fim, tem-se a tabela “Pedido”, aqui é possível adicionar pratos e produtos para que possa ser criado um pedido. Este também recebe uma forma de pagamento e se deve ser entregue a domicílio ou não. Após ser alimentada com essas informações e vinculada a um cliente, é possível realizar um pedido com sucesso.

3.2.3 Diagrama do Banco de Dados

Segundo Heuser (2009), o diagrama de banco de dados é uma descrição real do banco de dados da aplicação, ou seja, ele é o responsável por detalhar quais tabelas o banco irá possuir, e quais os atributos de cada tabela. Dessa maneira, é possível os desenvolvedores terem uma visão mais ampla em relação ao sistema. A Figura 4 mostra o diagrama do banco de dados do SiGeR, com todas suas tabelas, atributos e relacionamentos.

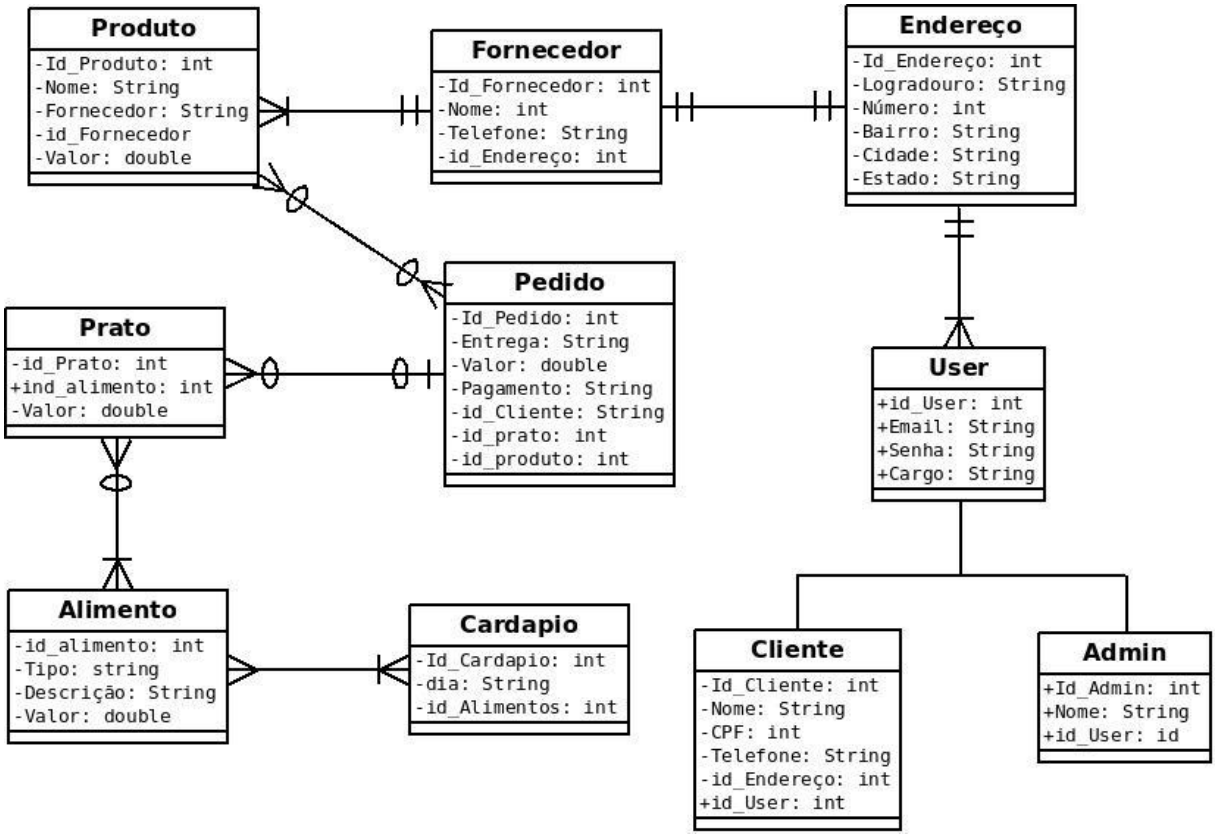


Figura 4 – Diagrama do Banco de Dados

No diagrama do banco de dados percebe-se a tabela "*User*" que possui os atributos Email e Senha que são responsáveis pelo *login* no sistema; o atributo Cargo é responsável por distinguir o usuário para que este tenha acesso apenas parte do sistema que o cabe; esta tabela se divide em outras duas, Cliente e Admin, com seus respectivos atributos relevantes para a aplicação. Tanto a tabela "User" como "Fornecedor" são diretamente ligadas a tabela "Endereço" que armazena as informações relacionadas ao endereço físico de cada um desses agentes. "Fornecedor" por sua vez tem ligação com a tabela "Produtos" que armazenam as

informações dos produtos que são revendidos no restaurante. A tabela “Alimento” guarda os dados de cada alimento oferecido no estabelecimento, como o tipo, descrição e valor de cada porção. Esta é ligada a “Cardápio” que atribuirá a cada alimento um dia da semana em que ele será vendido. Em “Prato” será armazenado todos os alimentos que o cliente pretende, e será atribuído um nome a esse prato e o valor total do mesmo será apresentado. Por fim tem-se a tabela “Pedido” que trará todas as informações do(s) prato(s) e/ou do(s) produto(s) que o cliente poderá adicionar em seu pedido, juntamente com o id do cliente, o valor total do pedido, a forma de pagamento e se deseja que o pedido seja entregue em sua residência ou não (*delivery*).

Com todas as informações apresentadas nesse capítulo é possível ter uma ideia bem clara do funcionamento do SiGeR e de como acontece a circulação de dados. Nos capítulos posteriores será possível ver o sistema funcionando na prática.

4 DEMONSTRAÇÃO DO SIGER

Nesse capítulo será possível enxergar os resultados atingidos ao utilizar as tecnologias e procedimentos citados nos capítulos anteriores. Aqui será demonstrado o SiGeR em sua versão atual, através de imagens de suas principais telas, o seu funcionamento na visão de seus usuários. Cada imagem será acompanhada de uma descrição que explicará o que acontece e como o usuário pode agir para ter sucesso em suas atividades. Como explicado na seção 2.1.4 o *design* responsivo será demonstrado através de imagens do sistema acessado através de um *smartphone*.

4.1 Principais Interfaces

Para utilizar as principais funções da aplicação o usuário deverá primeiramente realizar o *login* (Figura 5). Caso o usuário ainda não possua um *login* nessa tela ele pode pedir a solicitação de cadastro, onde ele preencherá um formulário com seus dados que será enviado para o administrador para aprovação. Ao entrar com seu *e-mail* e senha o usuário será redirecionado para a tela inicial Figura 6, que será diferente de acordo nível de acesso.

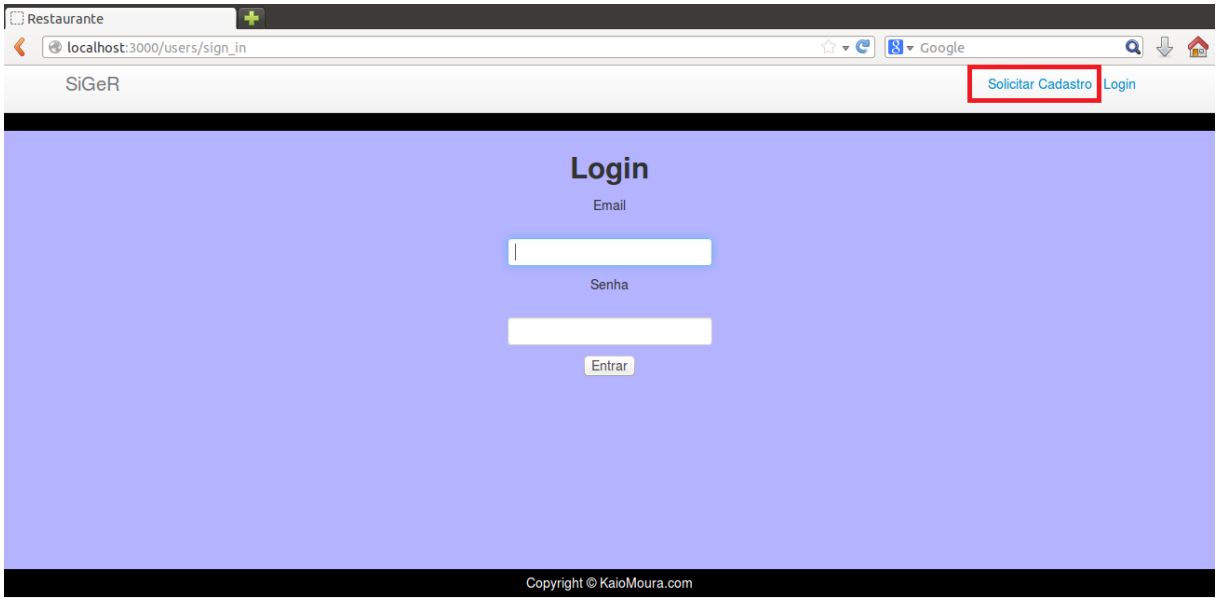


Figura 5 – Tela de *Login* do sistema acessado por um *notebook*

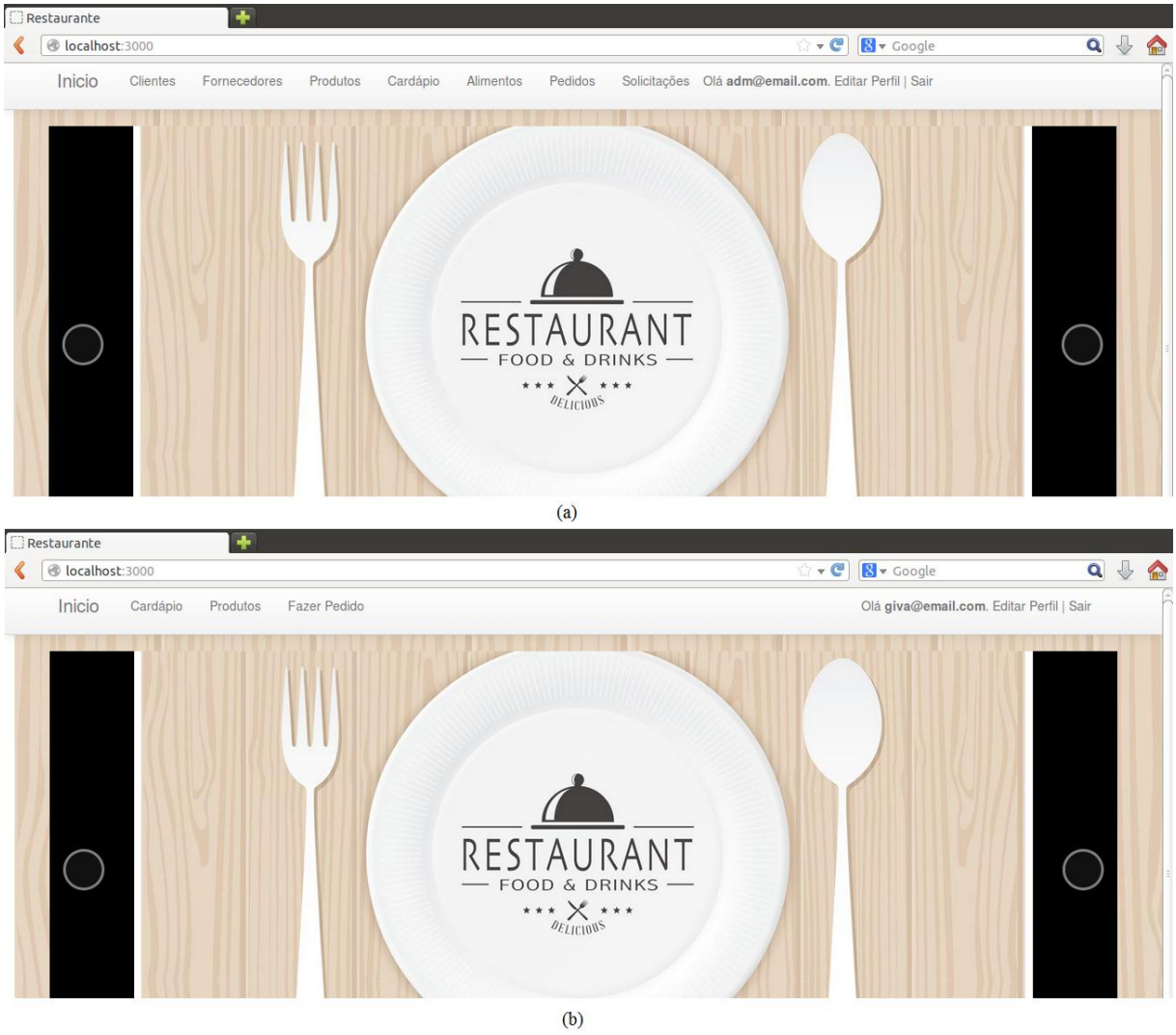


Figura 6 – Tela de Inicial acessada por um *notebook:* (a) usuário administrador; (b) usuário cliente

Apesar de serem visualmente semelhantes as telas inicias do sistema, tanto para um administrador quanto para um cliente comum, se diferem pela a quantidade de opções de atividades dispostas para cada usuário. Nota-se que as duas imagens se referem a tela de um *notebook* que devido ao seu tamanho permite que o usuário visualize todas as informações relevantes sem prejudicar a *interface* sobrecarregando-a. Na Figura 7 é apresentada a verão *mobile* do SiGeR acessada por um *smartphone*. Com a dimensão da tela é muito menor, o design responsivo permite adaptar as informações para que elas não sejam perdidas ou dispostas de forma indesejável ao usuário. Os menus presentes antes em uma barra principal são realocados em botão, Figura 7(a), que ao ser tocado permite que o usuário tenha acesso a aos seus respectivos links, Figura 7(b).

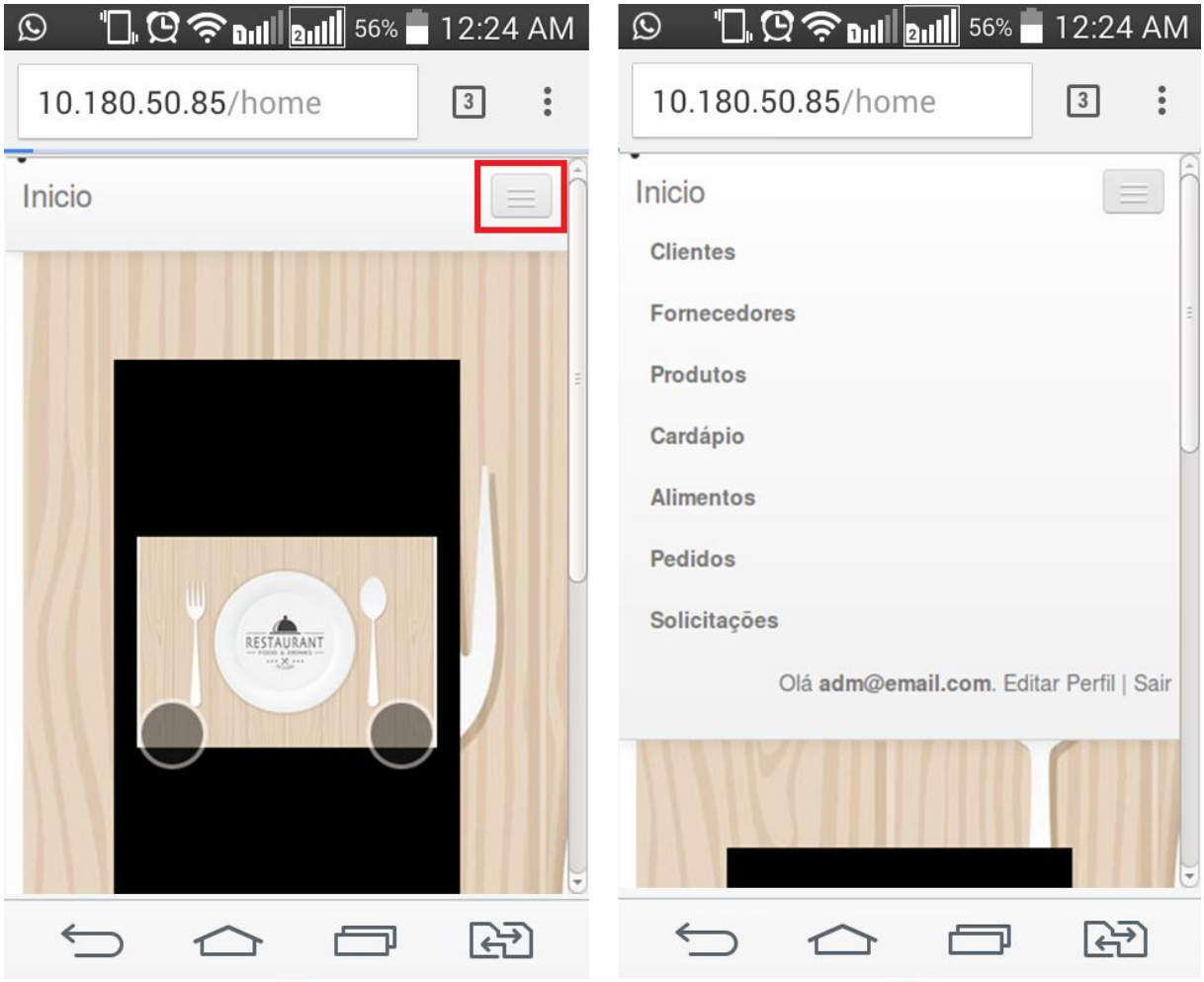
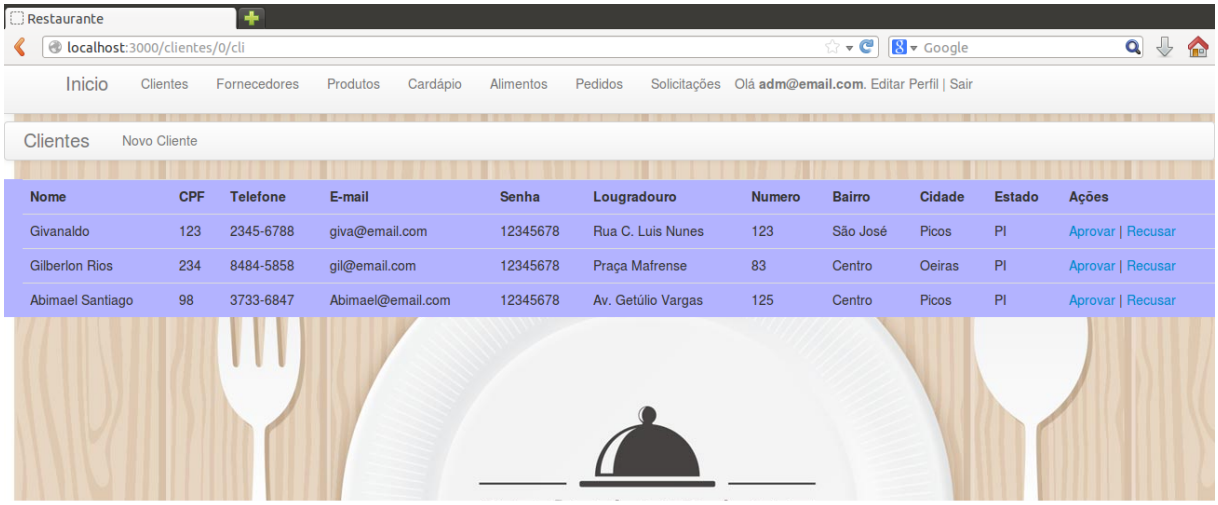


Figura 7 – Tela de Inicial acessada por um *smartphone:* (a) menus ocultos; (b) menus visíveis

Quando um novo usuário faz a solicitação de cadastro, o administrador recebe a solicitação através da página Solicitações. Nesta são listados todos os pedidos de cadastro onde o administrador poderá aceitar ou recusar essa solicitação de acordo com sua análise dos dados do possível cliente. A Figura 8(a) mostra a tela de solicitações. Quando o pedido é aprovado, esse usuário passa a integrar a lista de clientes, os quais ganham um *login* de acesso. A tela do sistema que mostra os clientes cadastrados está representada na Figura 8(b).



(a)

(b)

Figura 8 – Telas de Clientes acessadas por um *notebook:* (a) Solicitações; (b) Clientes Cadastrados.

Na Figura 9(a) é mostrada a tela de Fornecedores. Aqui são listados todos os fornecedores cadastrados no sistema com seus respectivos dados. Nessa mesma tela é possível criar um novo fornecedor, excluir e editar outros já cadastrados. Logo em seguida, Figura 9(b), podemos observar a tela de Produtos, onde são listados todos os produtos oferecidos pelo estabelecimento. Percebe-se que cada produto possui um fornecedor, este deve ser indicado sempre que for cadastrado um novo produto para a venda.

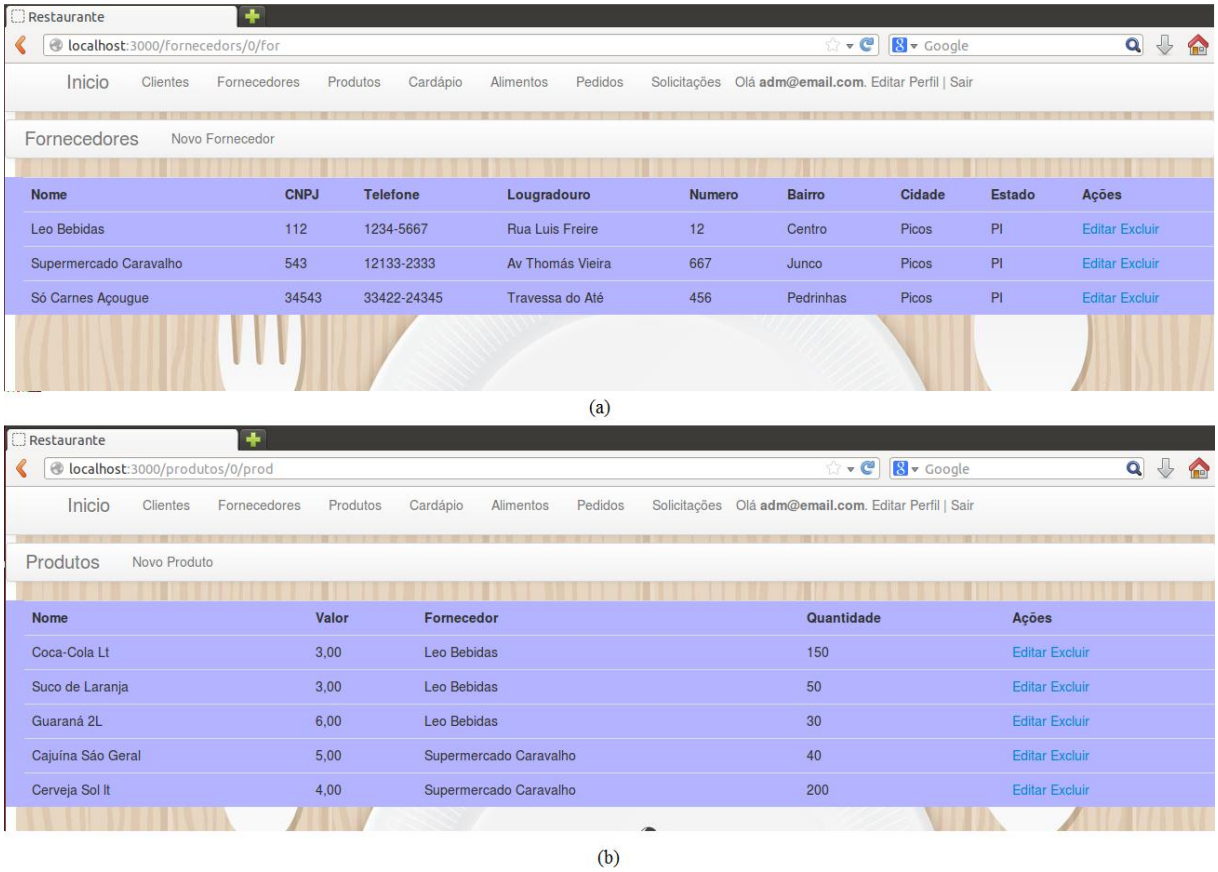


Figura 9 – Telas do Sistema acessada por um *notebook:* (a) Tela de Fornecedores; (b) Tela de Produtos.

Quando um cliente deseja fazer um pedido, ele deve clicar no menu “Fazer Pedido” onde será redirecionado para uma tela onde poderá montar o seu prato, Figura 10(a). Aqui ele nomeará o prato afim de identifica-lo e escolherá no cardápio os alimentos que deseja. Este cardápio é gerado a partir da lista de alimentos cadastrados para o dia da semana em que o pedido está sendo realizado. Após concluir o(s) prato(s), o cliente será redirecionado para a página onde poderá adicionar produtos ao seu pedido e escolher a forma de pagamento e de entrega, Figura 10(b).

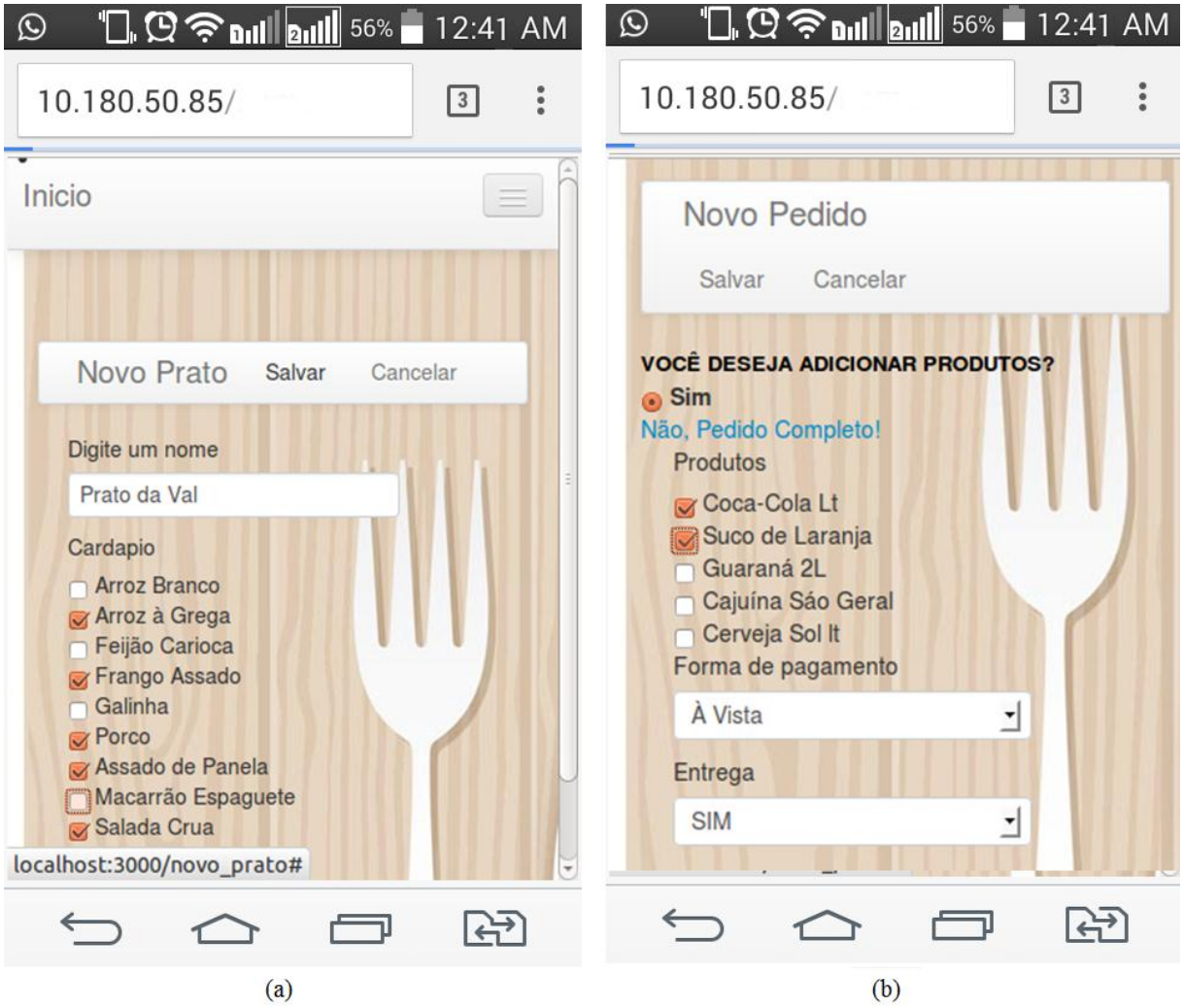


Figura 10 – Telas do Sistema acessadas por um *smartphone:* (a) Montar Prato; (b) Adicionar Produtos

Por fim, o cliente é redirecionado para a tela de fechamento do pedido (Figura 11). Nessa página será disposto o resumo do pedido, o número do mesmo; o valor total com todas as despesas; a forma de pagamento; e se será entregue a domicílio. Se tudo estiver de acordo com o que o cliente deseja, ele pode enfim concluir o pedido.

Figura 11 – Tela de Fechamento de Pedido acessada por um *notebook*

5 CONCLUSÕES E TRABALHOS FUTUROS

A implantação de um sistema de informação computacional em um organização nos dias atuais é praticamente indispensável. A quantidade de informações que circulam em um estabelecimento é enorme, e se não houver um bom sistema que as gerencie pode culminar até no fim da organização. O trabalho aqui apresentado possui exatamente o objetivo de evitar que que isso ocorra, de forma a melhorar e automatizar a gestão dessas informações. Desenvolvido na tecnologia web permite o acesso em qualquer lugar do planeta que possua uma conexão de rede.

Através da experiência no mercado e do estudo em estabelecimentos do ramo, foi possível traçar o levantamento de requisitos que iriam servir de base para a produção da aplicação. Observando os problemas e dificuldades pôde-se trabalhar em cima destes para que o sistema possa dar apoio aos usuários e melhorar a forma de trabalho. Posteriormente foi estudado as tecnologias que melhor se encaixavam para que juntas pudessem solucionar esses problemas. Após apresenta-las, foi possível dar uma olhada no resultado obtido, o próprio SiGeR. Através de imagens de telas do sistema, em diferentes dispositivos, observou-se o seu funcionamento e o passo-a-passo das principais atividades.

Como trabalhos futuros sugere-se, a criação de um frente de caixa que possibilitaria a gestão do setor financeiro do estabelecimento. Assim seria possível administrar as contas, as receitas, fluxo de caixa, etc; trabalhar na IHC da aplicação, procurando sempre melhorar a interação do usuário com o sistema; a utilização de técnicas de real time que permitiria a atualização das informações automáticas e em tempo real, além de possibilitar a implantação de um Chat que permitiria que os clientes pudessem se comunicar com funcionários e tirar dúvidas; a implantação do SiGeR em algum estabelecimento afim de realizar testes de funcionalidade e descobrir a aceitação dos usuários.

REFERÊNCIAS

BALDUINO, P. **Dominando JavaScript com jQuery.** São Paulo: Casa do Código, 2013.

CAELUM, Apostila curso**: Desenvolvimento Ágil para *Web* com Ruby on Rails, 2014**. Disponível em: < http://www.caelum.com.br/apostila-ruby-on-rails/ > Acesso em: 26 de julho de 2014.

DOCUMENTAÇÃO DO POSTGRESQL 8.0.0. **The PostgreSQL Global Development Group**. California: Copyright © 1996-2005 *The PostgreSQL Global Development Group.*

FUENTES, V. B. **Ruby on Rails**. Coloque sua aplicação web nos trilhos. São Paulo: Casa do Código, 2012.

GUEDES, G. T. A. **UML 2** Uma abordagem prática. 2ª Edição. São Paulo: Novatec, 2011.

HEUSER, C. A. **Projeto de Banco de Dados**. Bookman, Porto Alegre, Brasil, 2009.

LOPES, S. **A Web Mobile.** Programe para um mundo de muitos dispositivos. São Paulo: Casa do Código, 2013.

MENEGOTTO, A. B.; MIERLO, F. **A Linguagem Ruby**. UNISINOS - Universidade do Vale do Rio dos Sinos, 2002.

MULBERT, A. L.; AYRES, N. M. **Fundamentos para Sistemas de Informação.** Palhoça UnisulVirtual, 2005.

O'BRIEN, J. A. **Sistemas de informação e as decisões gerencias na era da internet**. 2 ed. São Paulo: Saraiva, 2006.

OLSINA, L.; LAFUENTE, G.; ROSSI, G. ***Specifying quality characteristics and atributes for websites. In: Web Engineering: Software Enginnering And Web Application Development***. Springer Berlin / Heidelberg, 2001.

PRESSMAN, R. S. **Engenharia de *Software***: uma abordagem profissional. 7 ed. Porto Alegre: AMGH, 2011.

REZENDE, D. A.; ABREU, A. F. **Tecnologia da informação aplicada a sistemas de informação empresariais**. 3. ed. São Paulo: Atlas, 2003.

REZENDE, D. A. **Engenharia de software e sistemas de informação**. Brasport, 2005.

RODRIGUES, A. **Desenvolvimento para Internet** Curitiba: LT, 2010.

SABBAGH, R. **SCRUM:** Gestão ágil para projetos de sucesso. São Paulo: Casa do Código, 2013.

SILVA, M. S. **Web Design Responsivo**. São Paulo: Novatec, 2014.

SOMMERVILLE, I. **Engenharia de *Software*.** 9 ed. São Paulo: Pearson Prentice Hall, 2011.

SOUZA, L**. Ruby**: Aprenda a programar na linguagem mais divertida. São Paulo: Casa do Código, 2012.

SPURLOCK, J. **Responsive Web Development Bootstrap**. 1 ed. Sebastopol: O'Reilly Media, 2013.

ZEMEL, T. **Web Design Responsivo**: Páginas adaptáveis para todos os dispositivos. São Paulo: Casa do Código, 2012.

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, KAIO CESAR MARCOS DE MOURA,

autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação DESENVOLVIMENTO DE UM SISTEMA WEB PARA CONTROLE E GERENCIAMENTO DE RESTAURANTES COM PRINCIPIOS DE RESPONSIVIDADE. de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI 05 de MARÇO de 2016.

Kaio César M. Moura

Assinatura

Kaio César M. Moura

Assinatura