UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JEFFERSON BARBOZA SANTANA

SISTEMA *WEB* PARA GERENCIAMENTO DE UMA METALÚRGICA

TRABALHO DE CONCLUSÃO DE CURSO

PATO BRANCO
2014

JEFFERSON BARBOZA SANTANA

SISTEMA *WEB* PARA GERENCIAMENTO DE UMA METALÚRGICA

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Câmpus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

Orientadora: Profª. Andreia Scariot Beulke

PATO BRANCO
2014

ATA Nº: **257**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **JEFFERSON BARBOZA SANTANA**.

Às 16:30 hrs do dia 18 de dezembro de 2014, Bloco V da UTFPR, Câmpus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Andréia Scariot Beulke (Orientadora), Beatriz Terezinha Borsoi (Convidada) e Soelaine Rodrigues Ascari (Convidada), para avaliar o Trabalho de Diplomação do aluno Jefferson Barboza Santana, matrícula 1116703, sob o título **Sistema Web para Gerenciamento de uma Metalúrgica**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, COADS. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 17:05 hrs foi encerrada a sessão.

Profa. Andréia Scariot Beulke, Esp.
Orientadora

Profa. Beatriz Terezinha Borsoi, Dr.
Convidada

Profa. Soelaine Rodrigues Ascari, M.Sc.
Convidada

Profa. Eliane Maria de Bortoli Fávero, M.Sc.
Coordenadora do Trabalho de Diplomação

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

AGRADECIMENTOS

Agradeço a Deus por ser em todos os momentos a principal Força e Esperança nos momentos de dificuldades.

A minha esposa e filhos, pelo carinho, paciência, incentivo e compreensão que tiveram comigo no desenvolvimento deste trabalho, onde por inúmeras vezes obriguei-me a trocar seus aconchegos pelo teclado do computador.

A minha orientadora, Professora Andreia Scariot Beulke, pelo enorme auxílio no desenvolvimento deste trabalho. Sugestionando as melhorias necessárias para se tornar um trabalho digno de aprovação.

A todos os professores pelo conhecimento repassado, dedicação e ensinamentos disponibilizados nas aulas, pois cada um de forma especial contribuiu para a conclusão desse trabalho e minha consequente formação profissional.

Agradeço também aos meus familiares pelo que me ensinaram e a meus colegas de faculdade e amigos, em especial a uma grande amiga que me ajudou muito nos momentos de desalento, Adriana Ariati.

Finalmente, agradeço a todos que de alguma forma contribuíram para elaboração deste Trabalho.

RESUMO

SANTANA, Jefferson Barboza. Sistema *Web* para gerenciamento de uma metalúrgica. 2014. 66 f. Monografia de trabalho de Conclusão de Curso - Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Universidade Tecnológica Federal do Paraná. Pato Branco, 2014.

As Tecnologias da Informação (TI) vêm sendo cada vez mais exploradas para fornecer suporte às atividades realizadas nas empresas. A Internet também trouxe mudanças significativas no mundo empresarial e se tornou um meio importante para potencializar negócios, pois aproxima consumidores e as empresas instantaneamente. Essa conexão abre novas oportunidades de negócios para as empresas. Um sistema de informação deve ser construído a partir de um modelo de negócio para que efetivamente atenda as necessidades dos usuários. Esse trabalho apresenta a modelagem e a implementação de um sistema para gerenciamento dos serviços oferecidos por uma indústria metalúrgica de pequeno porte como, por exemplo, realização de orçamentos, cadastros de clientes e produtos. Um sistema *Web* para gerenciamento de uma metalúrgica contribui de modo significativo no controle de matérias-primas e venda de produtos manufaturados assim como na realização de orçamentos. A modelagem também viabiliza os processos e as tecnologias necessárias para solução do problema. Para o desenvolvimento do sistema foi utilizada a linguagem Hypertext PreProcessor (PHP). A ferramenta WorkBench foi utilizada para a modelagem e gerenciamento do banco de dados e o MySQL para o banco de dados.

Palavras-chave: Sistema de informação para metalúrgica. Sistema *Web*. Linguagem PHP.

ABSTRACT

SANTANA, Jefferson Barboza. *Web* system for managing a metallurgical. 2014. 66 f. Monograph Working End of Course - Curso de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná. Pato Branco, 2014.

Information technologies (IT) have been increasingly exploited to provide support to the activities of the companies. The Internet has also brought significant changes in the business world and has become an important means to boost business as it nears consumers and businesses instantly. This connection opens new business opportunities for companies. An information system should be constructed from a business model that effectively meets the needs of users. This work presents the modeling and implementation of a system for management of the services offered by a small metal industry, for example, achievement of budgets, customer records and products. A Web system for managing a metallurgical contributes significantly in the raw materials control and sale of manufactured products as well as the realization of budgets. The modeling also enables the processes and technologies needed to solve the problem. For the development of the system was used to Hypertext Preprocessor language (PHP). The WorkBench tool was used for modeling and database management and MySQL for the database.

Keywords: Information system for metallurgy. Web system. PHP Language.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTAGENS DE CÓDIGO

LISTA DE SIGLAS

API	*Application Programming Interface*
COBRA	*The Common Object Request Broker Architecture*
CSS	*Cascading Style Sheets*
EIS	Sistemas de Informações Executivas
FPDF	*Free Portable Document Format*
GUI	*Graphical User Interface*
HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
IDL	*Interactive Data Language*
IMAP	*Internet Message Application Protocol*
JS	*Java Script*
MYSQl	*My Structured Query Language*
MVC	*Model View Controller*
PHP	*Personal Home Page*
POP3	*Post Office Protocol*
SGBD	Sistema Gerenciador de Banco de Dados
SI	Sistema de Informação
SIAE	Sistemas de Automação de Escritório
SIG	Sistemas de Informação Gerencial
SIPT	Sistemas de Processamento de Transações
SITE	Sistemas de Informação de Tarefas Especializadas
SMTP	*Simple Mail Transfer Protocol*
SQL	*Structured Query Language*
SSTD	Sistemas de Informação de Suporte a Tomada de Decisão
SYSML	*Systems Modeling Language*
TI	Tecnologia da Informação
UML	*Unified Modeling Language*
XML	*Extensible Markup Languege*

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo são apresentadas as considerações iniciais contendo o contexto no qual se insere a proposta deste trabalho, que é o desenvolvimento de um sistema *Web* para gerenciamento de uma indústria metalúrgica de pequeno porte. Também são apresentados os objetivos e a justificativa do trabalho. Por fim está a organização do texto.

1.1 CONSIDERAÇÕES INICIAIS

Em ambientes organizacionais o acesso facilitado aos recursos computacionais pode impactar significativamente em diversos aspectos, que vão do aumento da efetividade do trabalho até a geração de vantagens competitivas com a melhora do desempenho organizacional, além de possibilitar a geração de novos negócios.

Desta forma ao se desenvolver um *software* para uma empresa de pequeno porte, é de extrema importância que seus requisitos, sendo eles funcionais, não funcionais ou de infraestrutura sejam atendidos. Para tanto, é necessário que a modelagem do sistema em questão seja realizada de forma ordenada e padronizada, para que possa ser desenvolvido um *software* de qualidade e no tempo planejado, atendendo deste modo, as expectativas do mercado e dos usuários.

Os Sistemas de Informação (SI) são responsáveis por coletar, processar e gerar dados para suporte às atividades de uma organização e/ou empresa. As informações são apresentadas para o usuário a partir da entrada dos dados, que são posteriormente processados e geram o resultado que o usuário necessita para realizar suas tarefas com maior rapidez e eficiência. Um sistema de informação tem a finalidade de oferecer suporte às atividades nas empresas em nível operacional, gerencial ou estratégico (LAUDON; LAUDON, 2001).

A quantidade de dados (agregados em informação quando fornecem suporte às atividades de negócio) manipulada pelas empresas é cada vez maior. É preciso controle apropriado de tais dados incluindo receitas e despesas, para que lucros possam ser maximizados e custos reduzidos. Além da lucratividade, o gerenciamento adequado do negócio é relevante para conquistar novos clientes e

manter os existentes. A utilização de *softwares* auxilia na definição de margens próprias de lucros, realização de promoções e descontos.

Como forma de facilitar o acesso dos usuários, sejam funcionários ou clientes, as empresas vêm adequando seus sistemas para acesso via Internet. A *Web* permite flexibilidade de acesso remoto, possibilitando aos usuários executar algumas funções que o sistema disponibiliza sem a necessidade de presença no espaço físico da empresa.

Apesar de existir diversos sistemas para gerenciamento de negócios que podem ser obtidos de forma gratuita, verificou-se a oportunidade de desenvolver um sistema para uma indústria metalúrgica de pequeno porte. O desenvolvimento desse sistema é uma forma de estudo para o autor deste trabalho (pela realização de atividades de todo o ciclo de vida de desenvolvimento de *software*) e de atender as necessidades específicas da empresa para a qual o sistema foi desenvolvido.

1.2 OBJETIVOS

O objetivo geral se refere à contribuição principal deste trabalho. E os objetivos específicos o complementam.

1.2.1 Objetivo Geral

Implementar um sistema que possibilite o gerenciamento de uma indústria metalúrgica de pequeno porte.

1.2.2 Objetivos Específicos

- Agilizar o atendimento dos clientes de uma metalúrgica;
- Facilitar o processo de elaboração dos orçamentos para a empresa e para o cliente;
- Agilizar os processos de pedidos de orçamentos e a geração de relatórios.

1.3 JUSTIFICATIVA

O ambiente empresarial e a forma de gestão vêm sofrendo mudanças devido a alterações de valores e da competitividade. A TI vêm sendo cada vez mais exploradas para oferecer suporte às atividades realizadas diariamente nas empresas, como por exemplo, controle de entrada e saída de produtos, realização de orçamentos, cadastros de clientes, entre outras. Nesse contexto, verifica-se que as informações em conjunto com os recursos tecnológicos são essenciais para o suporte e o funcionamento estratégico e operacional das empresas. Para Carvalho (2012), a informação é um conjunto de dados que auxilia as pessoas a desenvolverem melhor suas atividades e na tomada de decisões pelo gestor em uma organização.

A Internet trouxe mudanças significativas no mundo empresarial e se tornou um meio importante para potencializar negócios, pois aproxima consumidores e as empresas instantaneamente. Essa conexão abre novas oportunidades de negócios para as empresas.

Um sistema *Web* para gerenciamento de uma metalúrgica contribui de modo significativo no auxílio à venda de produtos manufaturados assim como no controle de materiais e realização de orçamentos pelo cliente. Como o cliente é o principal ator de uma empresa, é importante oferecer um atendimento ágil e satisfatório, estabelecendo uma relação favorável entre custo, qualidade e tempo. Uma das funcionalidades do sistema implementado permite que o cliente realize orçamentos *online*. A capacidade de comunicação que a Internet oferece estabelece uma relação mais próxima entre a empresa e o cliente e faz com que a empresa entenda as necessidades do cliente para melhor atendê-lo.

Para o desenvolvimento deste sistema foi utilizada a arquitetura MVC (acrônimo de *Model View Controller*), a qual visa a divisão de classes de um sistema em camadas, proporcionando uma melhor visualização das classes de negócio e de sistema. A primeira camada compreende as classes de negócio (*Model*), a segunda as classes de controle (*Controller*) e a última é a camada de visão (*View*).

Para a modelagem do sistema foi utilizada a MySQL WorkBench que é uma ferramenta de *designer* visual ao banco de dados. A linguagem PHP (*Personal Home Page*), foi utilizada para o desenvolvimento do módulo *Web*. A escolha dessa

linguagem deve-se ao fato de ser *open source* e pelos recursos que a mesma oferece, como por exemplo, portabilidade, velocidade e robustez.

1.4 ESTRUTURA DO TRABALHO

Este texto está organizado em capítulos, este é o primeiro e apresenta a introdução com a visão geral do que será modelado como resultado da realização deste trabalho. No Capítulo 2 está o referencial teórico, centrado no conceito e tipos de sistemas de informação, arquitetura cliente e servidor, nos diagramas da UML (*Unified Modeling Language*) e padrão MVC. O Capítulo 3 apresenta os materiais e o método utilizados na modelagem do sistema. No Capítulo 4 é apresentado o resultado obtido com a realização deste trabalho que é o desenvolvimento do sistema. No Capítulo 5 está a conclusão e as perspectivas futuras para continuidade do trabalho.

2 REFERENCIAL TEÓRICO

Este capítulo apresenta o referencial teórico e abrange os Sistemas de Informação (SI) aplicados nas organizações e padrão MVC.

2.1 SISTEMAS DE INFORMAÇÃO

Laudon e Laudon (1999. p. 4) argumentam que Sistemas de Informação

> "podem ser definidos tecnicamente como um conjunto de componentes inter-relacionados que coletam, processam, armazenam e distribuem informação com a finalidade de dar suporte à tomada de decisões e controle em uma organização. Também os sistemas de informação podem auxiliar gerentes e trabalhadores a analisar problemas, a visualizar formas complexas e a criar novos produtos".

Além disso, os autores destacam que os SI podem ser definidos como uma solução organizacional e gerencial, baseada em tecnologia da informação, em resposta a um desafio apresentado pelo meio ambiente. Desta forma, há um incremento no papel da organização como um todo no planejamento de SI apresentando assim uma solução ou parte dela em um problema real, imposto pelo ambiente em que a empresa trabalha.

De acordo com Freitas (1992, p. 38), "a informação é o produto da análise dos dados existentes nas empresas que transmite conhecimento e pode auxiliar o executivo na tomada de decisão". Dalfovo (2000) explica que o uso eficaz da informação nas organizações passa a ser um patrimônio, que é considerado um fator chave para o sucesso das organizações. Este fator torna-se mais expressivo quando as organizações se defrontam com as mudanças de mercado e avanços das tecnologias.

O maior desafio que as empresas enfrentam atualmente é o de antecipar problemas e encontrar as soluções práticas, a fim de realizar os anseios objetivados por ela. Os SI têm um papel fundamental e cada vez maior em todas as organizações de negócios.

Assim, os SI devem ser elaborados visando sempre atender às necessidades dos tomadores de decisão, proporcionando-lhes suporte as suas necessidades. E neste sentido acredita-se que deva ser estruturado de forma a não permitir que se

trabalhe com dados de pouca ou nenhuma importância em detrimento de informações vitais.

Rodrigues (1996) explica que, sem se preocupar com o histórico da evolução dos Sistemas de Informação, pode-se dizer que, a partir de 1985, a informação passou a ser utilizada, mais orientadamente, como recurso estratégico.

A partir desta época, os SI começaram a ser vistos como mercadoria pelo sentido e papel a eles atribuídos pelas empresas. Stair (1999) conceitua sistemas como um conjunto de elementos ou componentes que interagem para atingir objetivos. Os próprios elementos e as relações entre eles determinam como o sistema trabalha. Os sistemas têm entradas, mecanismos de processamento, saídas e *feedback*. Bons sistemas ajudarão uma organização a atingir seus objetivos, aperfeiçoando os processos empresariais e adicionando valor aos seus produtos.

De acordo com Dalfovo (2000), sistema é um conjunto de partes independentes que, juntas, formam um todo, para exercer uma dada função. Os componentes de um sistema são as entradas, o processamento e saídas, visualizados conforme Figura 1.

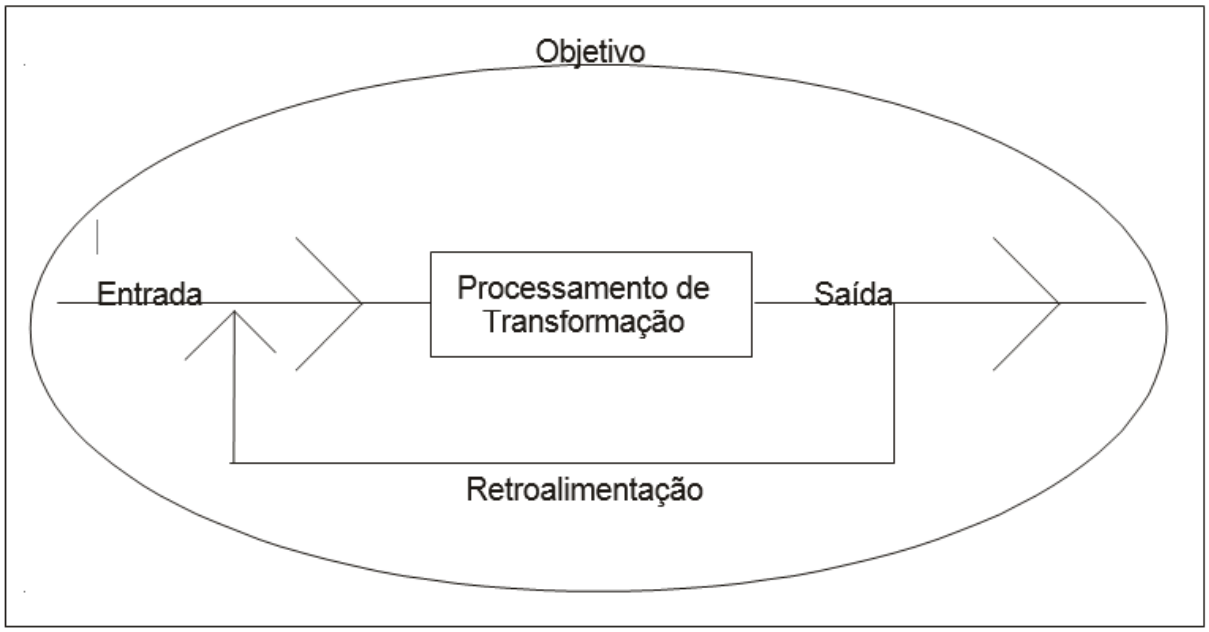

Figura 1 - Atividades de um sistema de informação
Fonte: Adaptado de Dalfovo (2000, p. 17).

Segundo Stair (1999) SI é um tipo especializado de sistema composto por uma série de elementos ou componentes inter-relacionados que coletam (entrada), manipulam e armazenam (processo), disseminam (saída) os dados e informações e fornecem um sistema de *feedback*. Essas atividades descritas a seguir:

a) entrada: é a atividade de captar e juntar dados primários. Independentemente do sistema envolvido, o tipo de entrada é determinado pela saída desejada do sistema. Esta entrada pode ser obtida por meio de um processo manual ou automatizado.

b) processo: envolve a conversão ou transformação dos dados em saídas. O processamento pode envolver cálculos, comparações e tomada de ações alternativas, além da armazenagem dos dados para uso futuro.

c) saída: envolve a produção de informações úteis, geralmente na forma de documentos, relatórios e dados de transações. As saídas podem incluir orçamento para o cliente, relatórios de vendas, de contas a pagar e a receber, os itens faltantes para fabricação de determinado orçamento, entre outros.

d) feedback: é uma saída usada para fazer ajustes ou modificações nas atividades de entrada ou processamento. Por exemplo, erros nas medidas propostas podem representar perda nos lucros ou de clientes para a concorrência se o processo não for modificado.

2.2 TIPOS DE SISTEMAS DE INFORMAÇÃO

Conforme Rezende (2003), SI foram divididos de acordo com as funções administrativas, que em decorrência de suas características geram sistemas distintos e funcionalidades específicas para ajudar os executivos nos vários níveis hierárquicos a tomarem decisões. A seguir é apresentado cada um deles:

a) Sistema de Informação Gerencial (SIG): segundo Oliveira (1998), SIG é o processo de transformação de dados em informações que são utilizadas na estrutura decisória da empresa, bem como proporciona a sustentação administrativa para aperfeiçoar os resultados esperados. Esse tipo de sistema é orientado para a tomada de decisões estruturadas.

b) Sistemas de Informação de Suporte a Tomada de Decisão (SSTD): de acordo com Dalfovo (1998), SSTD são sistemas voltados para administradores, especialistas, analistas e tomadores de decisão. São sistemas de acesso rápido, interativos, orientados para ação imediata.

c) Sistemas de Informação de Tarefas Especializadas (SITE): esses sistemas tornam o conhecimento de especialistas disponíveis para leigos, auxiliam a solução de problemas em áreas nas quais há necessidade de especialistas.

Conforme Dalfovo (1998), este tipo de sistema pode guiar o processo decisório e assegurar que os fatores de sucesso estejam considerados e auxiliam na tomada de decisões consistentes.

d) Sistema de Informação para Automação de Escritórios (SIAE): são sistemas que auxiliam no processamento de documentos e mensagens, fornecendo ferramentas que tornam o trabalho de escritório mais eficiente e efetivo. De acordo com a sua estrutura podem definir os formatos ou métodos das tarefas diárias, raramente afetando o conteúdo da informação. Fornecem ferramentas e métodos que podem ser usados de forma consistente. Os usuários são funcionários de escritório (STAIR, 1998).

e) Sistema de Processamento de Transações (SIPT): são sistemas que coletam e armazenam informações sobre transações. Suas estruturas basicamente implementam procedimentos e padrões para assegurar uma manutenção dos dados consistente e a tomada de decisão. Os seus usuários são pessoas que processam transações, como por exemplo, um funcionário do departamento de contas a receber (STAIR, 1998).

f) Sistemas de Informação para Executivos (EIS): de acordo com Furlan (1994), os EIS são sistemas voltados para os administradores com pouco, ou quase nenhum contato com Sistemas de Informação Automatizados. As principais características desse tipo de sistema são:

Gerar mapas, gráficos e dados que possam ser submetidos à análise estatística para suprir os executivos com informações comparativas, fáceis de entender.

Fornecer dados detalhados sobre passado, presente e tendências futuras das unidades de negócios em relação ao mercado para auxiliar o processo de planejamento e de controle da organização.

Possibilitar a análise das informações obtidas.

Permitir que o executivo se comunique com o mundo interno e externo através de interfaces amigáveis, que sejam flexíveis a ponto de se ajustarem ao seu estilo pessoal.

Oferecer ao executivo ferramentas de organização pessoal e de gerenciamento de projetos, tarefas e pessoas.

Informações de nível estratégico – indicadores de desempenho.

2.3 PADRÃO MVC

O padrão de projeto MVC surgiu em 1979 no projeto de Smalltalk na Xerox por Trygve Reenskaug e desde então vem sendo muito utilizado no desenvolvimento *Web*, pois ele permite aumentar o nível de abstração da aplicação e reduzir o nível de acoplamento entre os componentes. Este padrão determina que a aplicações devem ter uma separação em três níveis: *Model, View* e *Controller*.

Para Burbeck (1987), no paradigma MVC, o relacionamento com o usuário, a modelagem do domínio e a interação entre esses é feita de maneira distinta, por três tipos de objetos, cada um com uma tarefa especializada.

2.3.1 Características do Padrão MVC

O padrão de projeto MVC é estabelecido com uma estrutura dividida em três grupos de classes, sendo a primeira destas partes chamada *view*, que são as interfaces de comunicação com o usuário ou cliente, o *model* que são as classes onde se encontram os dados pertinentes a aplicação. Estas classes vão ter suas propriedades alteradas de acordo com o desenvolvimento do projeto e as necessidades de negócio que deverá atender. As classes do tipo *controller*, para Sampaio (2007), recebem a ação do usuário na *view*, atualizam as classes do tipo *model* de acordo com a solicitação enviada pelo usuário, recebe uma resposta desta manipulação realizada nestas classes e retorna este resultado para o usuário, por meio da *view*.

a) View

Na *view* está contida toda a apresentação da aplicação, ela faz requisições de dados ao modelo e apresenta os dados representados pelo *model*. Todas as saídas apresentas pelo sistema são visualizadas por meio da *view*. A classe *view* faz fronteira direta com o usuário final.

De acordo com Reenskaug (1979) as visões são representações das classes de modelo, que permitem a realização das operações das classes do tipo modelo. As visões são as interfaces com o usuário. É através da visão que o usuário interage com o sistema, e recebe as respostas de suas solicitações.

b) *Model*

O *model* ou modelo é o domínio específico em que a aplicação simula ou implementa. Nele fica encapsulada toda a lógica da aplicação. São responsabilidades do *model*: encapsular o estado da aplicação; responder à consulta sobre seu estado; expor as funcionalidades da aplicação; notificar as visões registradas sobre mudanças.

A classe *model*, também chamada de classe de negócio ou entidade, é uma classe que agrupa informações sobre o domínio do *software* em desenvolvimento, ela tem como propriedade informações e operações necessárias para a realização dos objetivos do *software*.

Para Bezerra (2007, p. 142), as classes de modelo “normalmente servem como um repositório para algumas informações do sistema. Nessas classes são alocadas as responsabilidades mais importantes do sistema”.

c) *Controller*

O *controller* ou controlador atua como interface realizando a associação do *model* com a *view*. A classe do tipo *controller* faz a intermediação do relacionamento entre classes do tipo visão e classes do tipo modelo. Realiza ainda a comunicação entre a view e o *model*, evitando o alto acoplamento das classes, e gerenciando a comunicação entre as mesmas.

De acordo com Bezerra (2007), as controladoras decidem o que o sistema fará diante de uma solicitação do usuário, gerenciando os demais objetos para a realização de uma determinada funcionalidade do sistema. As classes do tipo *controller* são consideradas classes de sistema. As controladoras existem para que a aplicação possa ser implementada. Não há como implementar um *software* utilizando apenas classes de negócio.

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais e os métodos utilizados, aqueles se referem às tecnologias e ferramentas utilizadas para modelar e implementar o sistema, estes reportam a sequência das principais atividades realizadas para desenvolver este trabalho.

3.1 MATERIAIS

Foram utilizadas as seguintes tecnologias e ferramentas para a modelagem e a implementação do sistema:

Ferramenta / Tecnologia	**Versão**	**Finalidade**	**Referência**
Visual Paradigm	10.1	Documentação da modelagem baseada na UML e ER do banco de dados	http://www.visual-paradigm.com/
NetBeans	7.3.1	IDE (*Integrated Development Environment*) de desenvolvimento	https://netbeans.org/features/index.html
PHP	5.4.7	Linguagem de programação	http://php.net/downloads.php
MySQL	5.6	Banco de dados	http://www.mysql.com/
MySQL WorkBench	5.2	Gerenciador do banco de dados	http://www.mysql.com/products/workbench/
Xampp Control Panel	3.1.0	Servidor *Web*	http://www.apachefriends.org/pt_br/xampp.html
Bootstrap	2.2.2	Framework de front-end	http://globocom.github.io/bootstrap/
Balsamiq Mockup	2.2.24	Permite criar maquete de *software*	http://balsamiq.com/download/

Quadro 1 - Apresentação das ferramentas e tecnologias

a) Visual Paradigm

Visual Paradigm for UML é uma ferramenta de modelagem para todos os tipos de diagramas UML (VISUAL PARADIGM, 2014). Essa ferramenta fornece suporte para gerenciamento de casos de uso, diagrama de requisitos SysML e

projeto de banco de dados com diagrama de entidades e relacionamentos. A Figura 2 apresenta um print screen da interface principal dessa ferramenta.

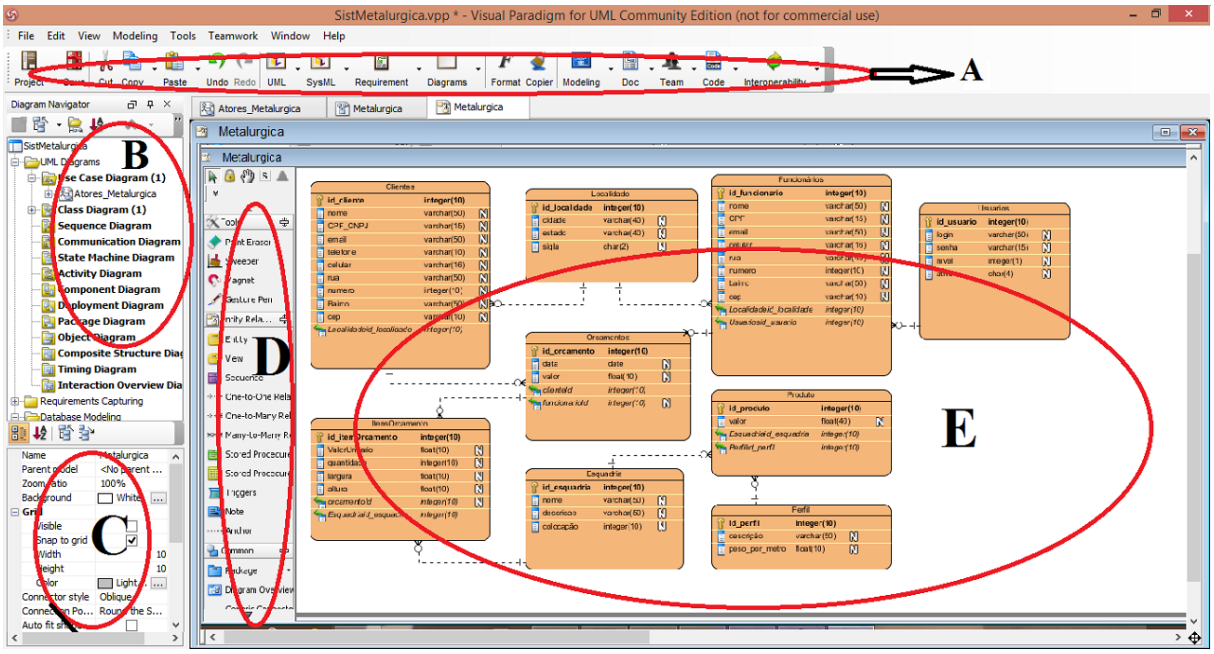

Figura 2 - Tela inicial da ferramenta Visual Paradigm

As partes destacadas na Figura 2 estão descritas a seguir:

A - Barra de ferramentas, com os atalhos para as principais funcionalidades do Visual Paradigm.

B - Navegador de diagramas, nesta área da interface são apresentados os diagramas armazenados. A disponibilização dos arquivos em pastas por tipo de diagrama facilita a localização dos mesmos e organização do projeto.

C - Apresenta as propriedades do elemento selecionado do diagrama em edição.

D - Ferramentas e componentes, área na qual ficam disponíveis os componentes utilizados para compor o diagrama e ferramentas relacionadas.

E - Área de edição, nesta área são criados graficamente os diagramas por meio dos elementos dispostos na lateral esquerda dessa área.

Visual Paradigm é compatível com a UML 2.0, permite a visualização, criação e edição de diagramas de fluxos de trabalho. A ferramenta possibilita a geração de código na linguagem Java a partir de diagramas. E engenharia reversa em Java, C++, XML (*Extensible Markup Language*) Schema, .NET e COBRA IDL (*The Common Object Request Broker Architecture Interactive Data Language*). Além de permitir a geração de códigos compatíveis com XML.

b) NetBeans

A IDE NetBeans (NETBEANS, 2014) é um ambiente de desenvolvimento multiplataforma e agrupa diversas funcionalidades necessárias para implementar um sistema. Ela possui um grande conjunto de bibliotecas, módulos e APIs (*Application Programming Interface*) que compõem, basicamente, um conjunto de rotinas, protocolos e ferramentas. Essa IDE possibilita desenvolver aplicativos para as plataformas *desktop*, *Web* e *mobile*. A Figura 3 apresenta sua tela principal.

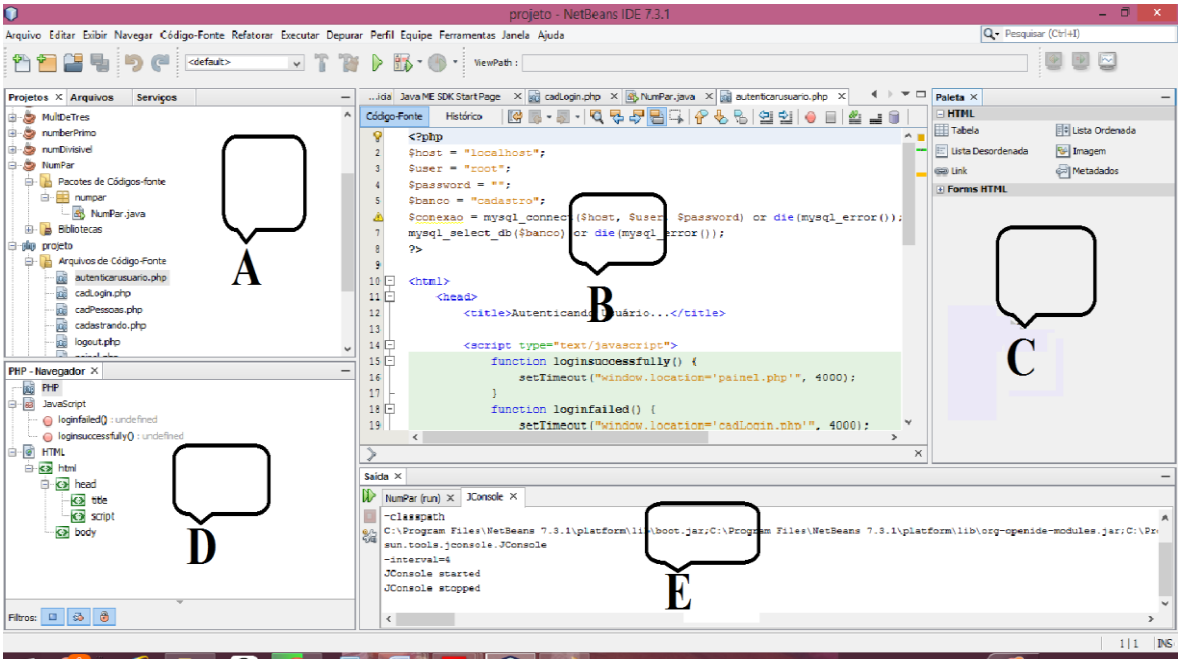

Figura 3 - Interface da ide Netbeans

As partes em destaque na Figura 3 indicam:

A - Local onde são apresentados os projetos, cuja visualização pode ser ativada selecionando a opção correspondente no menu Janela. Em Arquivos são apresentados os diretórios dos projetos, incluindo arquivos e pastas não mostrados na janela Projetos. Serviços é uma visão lógica de recursos tais como bancos de dados e servidores *Web* que estão registrados no ambiente.

B - Estão o editor de código e o *Graphical User Interface* (GUI) Construtor. GUI Construtor é local de desenho dos formulários GUI. Se em um projeto for criada uma classe GUI, por exemplo, na área do editor de código pode-se alternar entre o editor de código fonte (selecionando a guia código-fonte) e o GUI Construtor (selecionando a guia Projeto).

C - Nesta área está a paleta e a janela propriedades. Na paleta estão os

componentes disponíveis para diversos editores do IDE, tais como o GUI Construtor e o Visual Mobile Designer. Para formulários GUI existem contêineres e outros componentes visuais, tais como botões, rótulos, painéis, etc. Em propriedades são mostrados nomes e valores de propriedades de objetos que fazem parte de um projeto.

D - Contém o navegador e inspetor. Na primeira se pode navegar por diferentes partes do arquivo selecionado. A janela Inspetor mostra a hierarquia de todos os componentes contidos no formulário GUI aberto na área do Editor;

E - Composta por abas. A mais importante é a saída, que mostra resultados de execução de programas e erros de compilação. Em tarefas são mostradas coisas a fazer, principalmente nos arquivos dos projetos. Por exemplo, quando um novo método é criado usando o modelo, a IDE inclui um comentário. Esse comentário aparece em tarefas, alertando que é necessário inserir código naquela posição. Existem outras janelas que podem ser mostradas nessa área: resultados do teste e de pesquisas e ocorrências.

c) Linguagem PHP

PHP é uma linguagem de programação dinâmica para *Web* (PHP, 2011) que é processada no servidor, retornando para o cliente somente *HyperText Markup Language* (HTML). Desta forma o código fonte não é exposto e isso é importante na interação com banco de dados ou outros componentes que possuam informações sigilosas que precisem ser especificadas no código.

A linguagem PHP tem suporte a praticamente todos os bancos de dados existentes no mercado, o que torna simples a integração com aplicações que necessitem desta tecnologia. A linguagem também suporta outros protocolos como SMTP (*Simple Mail Transfer Protocol)*, POP3 (*Post Office Protocol)* e IMAP (*Internet Message Application Protocol*).

d) MySQL

O MySQL é um servidor e gerenciador de banco de dados relacional que utiliza a linguagem SQL (MYSQL, 2012). Ele é de licença dupla (*software* livre e paga), projetado inicialmente para trabalhar com aplicações de pequeno e médio portes, mas atualmente atende também aplicações de grande porte.

As principais características incorporadas na versão 5 do MySQL (MILANI, 2007) são:

- Visões: são consultas pré-programadas ao banco de dados que permitem unir duas ou mais tabelas e retornar uma única tabela como resultado. Com o uso de visões, operações frequentes com uniões de tabelas podem ser centralizadas. É possível também utilizá-las para controle de acesso às tabelas que fornecem dados para visões.

- Cursores: cursores possibilitam a navegação em conjuntos de resultados. E possível navegar pelos registros de uma tabela a partir de estruturas de repetição, permitindo realizar operações necessárias e transações para cada linha da tabela.

- *Information Schema*: são tabelas responsáveis pela organização dos recursos do banco de dados, conhecidos como dicionário de dados, ou metadados.

- Integridade referencial: os relacionamentos entre tabelas distintas são gerenciados pelo banco de dados quando de inclusão, alteração e exclusão.

- Clusterização: é baseada na integração e sincronismo de dois ou mais servidores que dividem a demanda de execuções entre si. Além da sincronização de um *cluster*, é possível especificar um balanceador de cargas. Esse recurso gerenciará o redirecionamento de servidores secundários e balanceará as consultas recebidas pelo *cluster*, distribuindo-as pelos servidores de acordo com sua disponibilidade.

e) MySQL Workbench

MySQL Workbench (WORKBENCH, 2011) é uma ferramenta gráfica para modelagem de dados. A ferramenta possibilita trabalhar diretamente com objetos *schema*, além de fazer a separação do modelo lógico do catálogo de banco de dados. Toda a criação dos relacionamentos entre as tabelas pode ser baseada em chaves estrangeiras. Outro recurso que a ferramenta possibilita é realizar a engenharia reversa de esquemas do banco de dados, bem como gerar os *scripts* em SQL.

Com essa ferramenta, a modelagem do banco de dados pode assumir níveis conceituais, lógicos e físicos. MySQL Workbench apresenta uma arquitetura extensível, sendo possível visualizar a representação de tabelas, funções, entre outros. A Figura 4 apresenta os três serviços disponíveis pelo MySQL Workbench.

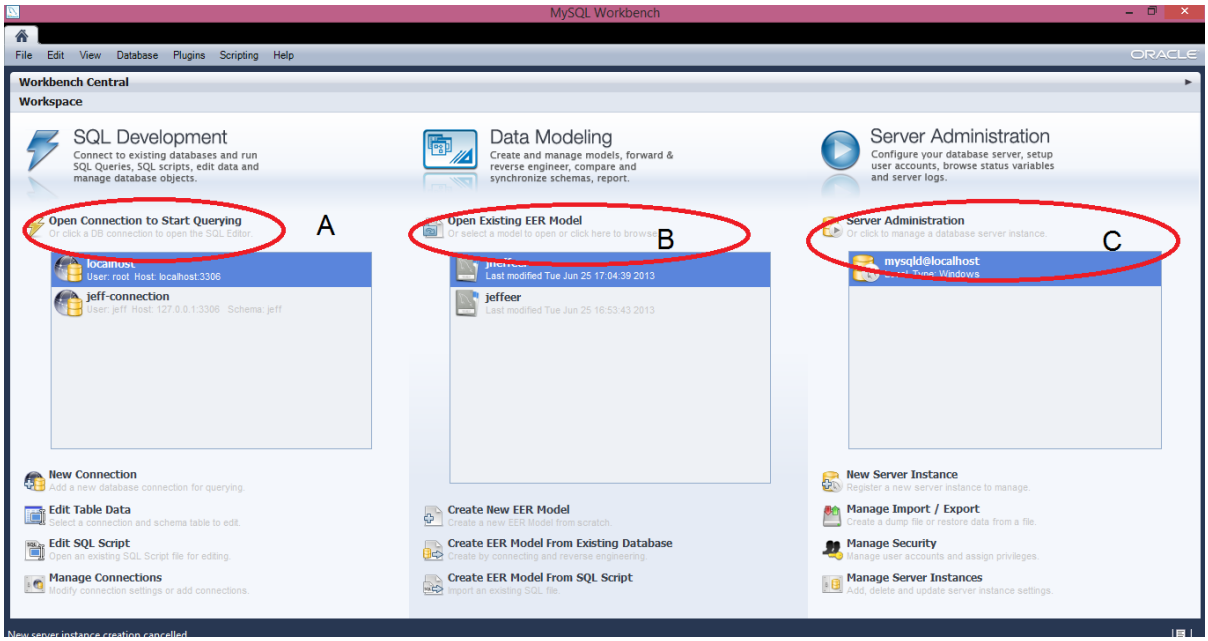

Figura 4 - Tela dos serviços do MySQL Workbench

As áreas destacadas da Figura são:

A - *SQL Development* é utilizado para conexão a bases de dados registradas no servidor possibilitando a execução de *query*, *scripts* SQL e editar dados dessas bases. O editor SQL informa o usuário de erros de sintaxe automaticamente.

B - *Data Modeling* possibilita ao usuário criar e gerenciar modelos de dados, executar engenharia reversa utilizando uma base de dados registrada para gerar um novo diagrama, gera os *scripts* completos ou apenas os que o usuário escolher para executar no *SQL Development*.

C - *Server Administration* contém as configurações gerais do servidor, manipulação de usuários, controle de portas e conexões do sistema e do servidor, *backups* e restaurações de bases de dados.

A área superior da tela destina-se à edição e modelagem dos dados. Para inserir uma tabela, basta clicar no botão "*The Table Tool*" e arrastá-lo para a página de edição. As tabelas criadas ficam disponíveis em uma aba central, sendo possível editar os atributos, colunas, chaves primárias e estrangeiras, entre outros. Após modelar é necessário clicar na opção "*Apply Changes*" para aplicar as alterações efetuadas.

f) Xampp Control Panel

O Apache é um servidor *Web* de código fonte aberto, configurável, compatível com linguagens dinâmicas como PHP e disponível para vários sistemas operacionais incluindo Windows e Unix sendo gratuito mesmo para uso comercial (APACHE, 2011).

g) Framework Bootstrap

O *framework* Bootstrap foi criado em 2011 por Mark Otto e Jacob Thomton do Twitter a partir de código que eles já usavam internamente, e liberado como *open source*, ganhando muitos adeptos. Tal projeto cresceu bastante em maturidade e importância no mercado a ponto de se desvincular do Twitter e ser apenas o Bootstrap.

Esse *framework* possui os seguintes recursos: reset CSS; estilo visual base pra maioria das *tags*; ícones; grids prontos para uso; componentes CSS; plugins JavaScript. Para usar o Bootstrap, é necessário incluir o arquivo externo de estilização na página que será utilizado.

Deste modo já traz alguns benefícios, primeiro ele é muito simples de usar, basta coloca-lo no código e usar. Segundo, depois de compilado, bootstrap contém nada mais que *Cascading Style Sheet* (CSS), não sendo necessários imagens adicionais, Flash ou Javascripts. Um reset é aplicado, e essas *tags* ganham estilo e tipografia base. Isso quer dizer que é possível usar *tags* como um marcador “h1” ou “p” e elas terão um estilo característico do Bootstrap.

O Bootstrap possui um *grid* pronto e várias classes. O *grid* do Bootstrap trabalha com a ideia de 12 colunas e pode escrever o código escolhendo quantas colunas serão ocupadas. Alguns exemplos são apresentados na Figura 5.

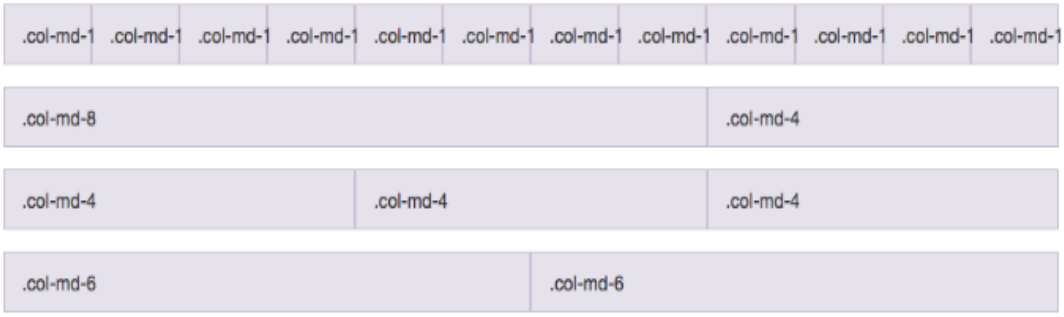

Figura 5 - Grid do Bootstrap

Essas classes de coluna são as que definem o tamanho de cada elemento na página com base nas 12 partes do grid padrão. Em código:

```
<div class="row">
        <div class="col-md-4">
        ...
        </div>
        <div class="col-md-8">
        ...
        </div>
</div>
```

No código anterior, para o *grid* funcionar, ao redor das colunas usa-se um div com *class row*. Os grids são responsivos podendo ser aplicados diferentes leiautes de colunas no código ao mesmo tempo e cada um deles será aplicado em determinada situação.

O Bootstrap possui diversas famílias de classes de *grids* prontas:

- col-xs- : *Extra small*. < 768px (tela menor que 768 pixeis);
- col-sm- : *Small* (*tablets*). >= 768px (tela maior ou igual 768 pixeis);
- col-md- : *Medium (desktops*). >= 992px (tela maior ou igual 992 pixeis);
- col-lg- : *Large (desktops*). >= 1200px (tela maior ou igual 1200 pixeis).

O Bootstrap possibilita aplicar mais de uma classe ao mesmo tempo no mesmo elemento:

```
<div class="row">
        <div class="col-xs-6 col-sm-4">
        ...
        </div>
        <div class="col-xs-6 col-sm-8">
        ...
        </div>
</div>
```

Nesse exemplo, o grid divide no meio (6 colunas para cada lado) em telas muito pequenas, mas depois divide em 4 e 8 para telas um pouco maiores.

h) Balsamiq Mockup

Balsamiq Mockups é uma ferramenta *wireframing* (desenho de maquete). Essa ferramenta reproduz a experiência de desenhar em um quadro branco, mas usando um computador e assim elaborar o protótipo de interface (interação) do *software*.

A ferramenta é composta por quatro blocos: a barra de aplicativos, a biblioteca de interface do usuário, a tela maquete e o navegador de arquivos. Cada uma conforme indicação na Figura 6.

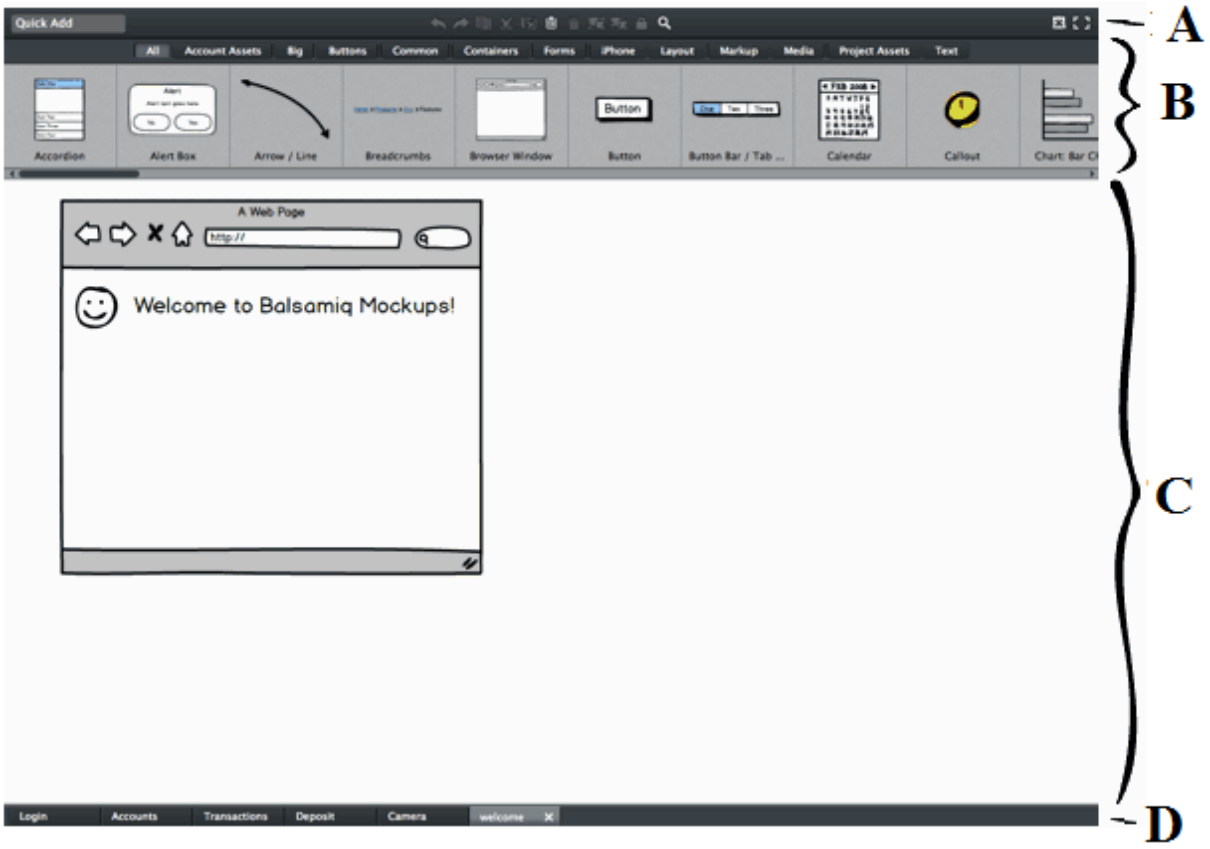

Figura 6 - Tela da ferramenta Balsimaq Mockup

A - A Barra de aplicação inclui menus, a ferramenta Add rápido e a barra de ferramentas. A ferramenta Quick Add é o caminho mais rápido para adicionar controles de interface do usuário para sua maquete. Para usar add rápido é necessário clicar dentro da caixa de entrada add rápido. Desse modo após digitar algumas letras do nome do controle de interface do usuário que deseja adicionar, o *Quick Add* apresenta uma lista de sugestões. A barra de ferramentas inclui a maioria dos comandos do menu edit: desfazer, refazer, duplicar, cortar, copiar, colar e excluir.

B – A Biblioteca ou Interface do Usuário se encontra logo abaixo da barra de aplicativos. Ela lista todos os diferentes tipos de controle de interface do usuário que Mockups suporta, em ordem alfabética. Um dos objetivos é permitir a inclusão de controles de interface do usuário para a tela maquete.

Para incluir um novo controle de interface de usuário para a tela, basta selecionar o tipo de controle que se deseja adicionar, em seguida "arrastar e soltar" para a tela Mockup. A Biblioteca de componentes pode ser posicionada no topo, à esquerda ou à direita da janela do aplicativo.

C - Tela da Maquete, ao adicionar controles de interface do usuário, é possível movê-los, redimensioná-los e ajustá-los. A tela Mockup aumenta e diminui com o navegador ou a janela do aplicativo.

D - Navegador de Arquivos, por padrão, o navegador de arquivos Mockups aparece na parte inferior da janela do aplicativo. Esta barra exibe os arquivos de maquetes aberto como uma série de abas e a maquete atual em uso permanece realçada. Quando as guias para o número de Mockups abertos não podem ser exibidos no espaço disponível na área de navegação de arquivos, um menu *pop-up* é exibido para navegar pelos arquivos.

i) Java Script

Desenvolvida pela Netscape, a linguagem JavaScript foi criada para trabalhar com aplicações interativas nas páginas HTML.A linguagem também é conhecida como uma extensão da linguagem HTML, pois os comandos JavaScript ao serem implementados nas páginas são interpretados pelo navegador sem ser necessário a compilação (JAVA, 2013).

O JavaScript suporta muitos comandos e sintaxes da linguagem Java. Uma das vantagens é que além de controlar dinamicamente o comportamento de objetos nas páginas, o retorno da linguagem é imediato, pois não é necessário recorrer ao servidor.

j) JQuery

JQuery é uma biblioteca JavaScript desenvolvida para simplificar os *scripts* que interagem com o HTML. Sua sintaxe foi desenvolvida para tornar mais simples a navegação do documento HTML e criar animações. A biblioteca oferece a possibilidade de criação de *plugins* sobre ela.

3.2 MÉTODO

As etapas para a modelagem e a implementação do sistema seguiram o modelo sequencial linear proposto por Pressman (2005). O uso desse modelo é justificado porque o sistema é simples e os requisitos do mesmo foram completamente definidos no início do processo.

As principais atividades realizadas para o desenvolvimento deste trabalho foram (PRESSMAN, 2005):

a) Levantamento de Requisitos

Os requisitos foram levantados e definidos tendo como base a necessidade de uma indústria metalúrgica de pequeno porte. O levantamento de requisitos foi realizado a partir da identificação das necessidades de uma empresa do ramo de metalurgia.

b) Planejamento

Inicialmente foi elaborada uma visão geral da página principal do sistema meio de um Balsamiq Mockups. Os requisitos levantados foram modelados por meio de diagrama de casos de uso e de classe. As tabelas e respectivos campos do banco de dados foram definidos a partir das classes identificadas como persistentes.

c) Desenvolvimento

A ferramenta NetBeans foi utilizada para implementar as funcionalidades do sistema utilizando a linguagem PHP e o banco de dados MySQL.

d) Testes

Os testes de navegabilidade, funcionalidade e compatibilidade foram realizados pelo autor deste trabalho.

4 SISTEMA DESENVOLVIDO

Neste capítulo são apresentados os resultados da realização do trabalho, que se refere à modelagem e à implementação de um sistema para gerenciamento das atividades de uma indústria metalúrgica de pequeno porte.

4.1 APRESENTAÇÃO DO SISTEMA

O sistema gerencia as atividades de uma indústria metalúrgica de pequeno porte. Os processos controlados incluem basicamente cadastros e a geração de orçamento das aberturas. O acesso dos vendedores ocorrerá por uma interface *Web*.

O sistema está centrado em cadastros (clientes, funcionários, produtos entre outros). Na tela de cadastro de produtos serão listados os produtos com suas respectivas quantidades.

No cadastro de usuário existe um controle básico de nível de acesso. O sistema terá três níveis de usuário:

a) Administrador - acesso a todas as funcionalidades do sistema;

b) Vendedor - com algumas restrições. Ao aplicar os dados nesta tabela de usuário, será também especificado qual o nível de acesso da pessoa cadastrada.

c) Cliente - acesso para realização de orçamentos. Sistema permitirá ao cliente gerar orçamentos *on-line*. Tipos de aberturas, medidas e custos, seguido pelo valor total do orçamento. O sistema possibilitará ao funcionário emitir relatório dos orçamentos gerados.

4.2 MODELAGEM DO SISTEMA

A modelagem do sistema foi realizada com base na *Unified Modeling Language* (UML). Booch, Rumbaugh e Jacobson (2000) se referem à UML como uma linguagem-padrão para a elaboração da estrutura de projetos de *software*. Essa linguagem poderá ser utilizada para visualização, especificação, construção e documentação de artefatos de *software*.

Os principais requisitos definidos para o sistema são:

1) O sistema permite que o cliente:

a) realize o seu orçamento.

b) consulte produtos.

2) O sistema deverá permitir que o administrador ou funcionário:

a) cadastre produtos.

b) edite produtos.

c) exclua produtos.

d) realize orçamentos.

3) O sistema deverá fornecer relatórios de:

a) orçamentos.

Inicialmente, uma representação visual simplificada do leiaute foi definida para o menu da metalúrgica. Essa representação (Figura 7) foi desenvolvida com a ferramenta Balsamiq Mockups.

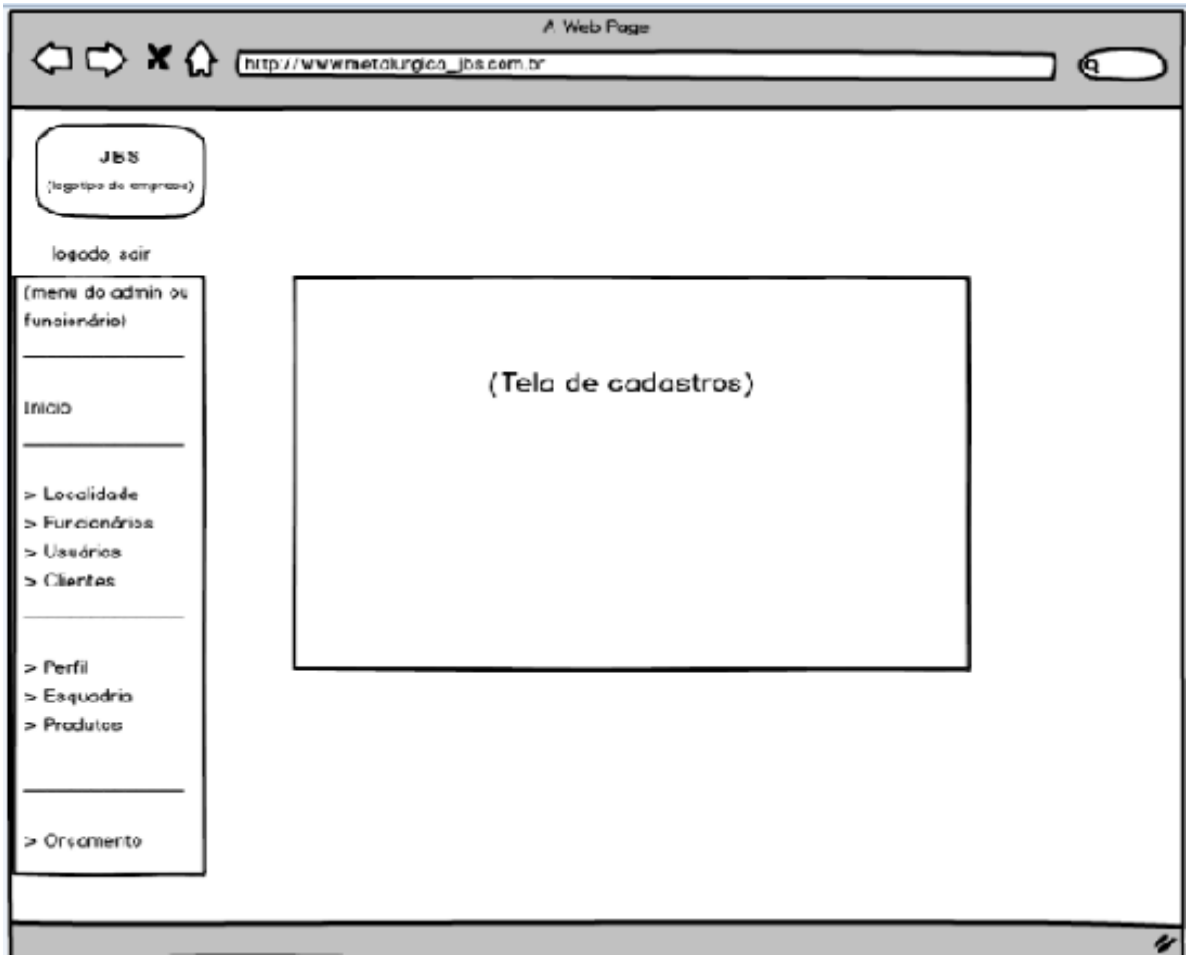

Figura 7 - Organização do leiaute do sistema

A Figura 8 apresenta o diagrama de casos de uso definidos para o sistema.

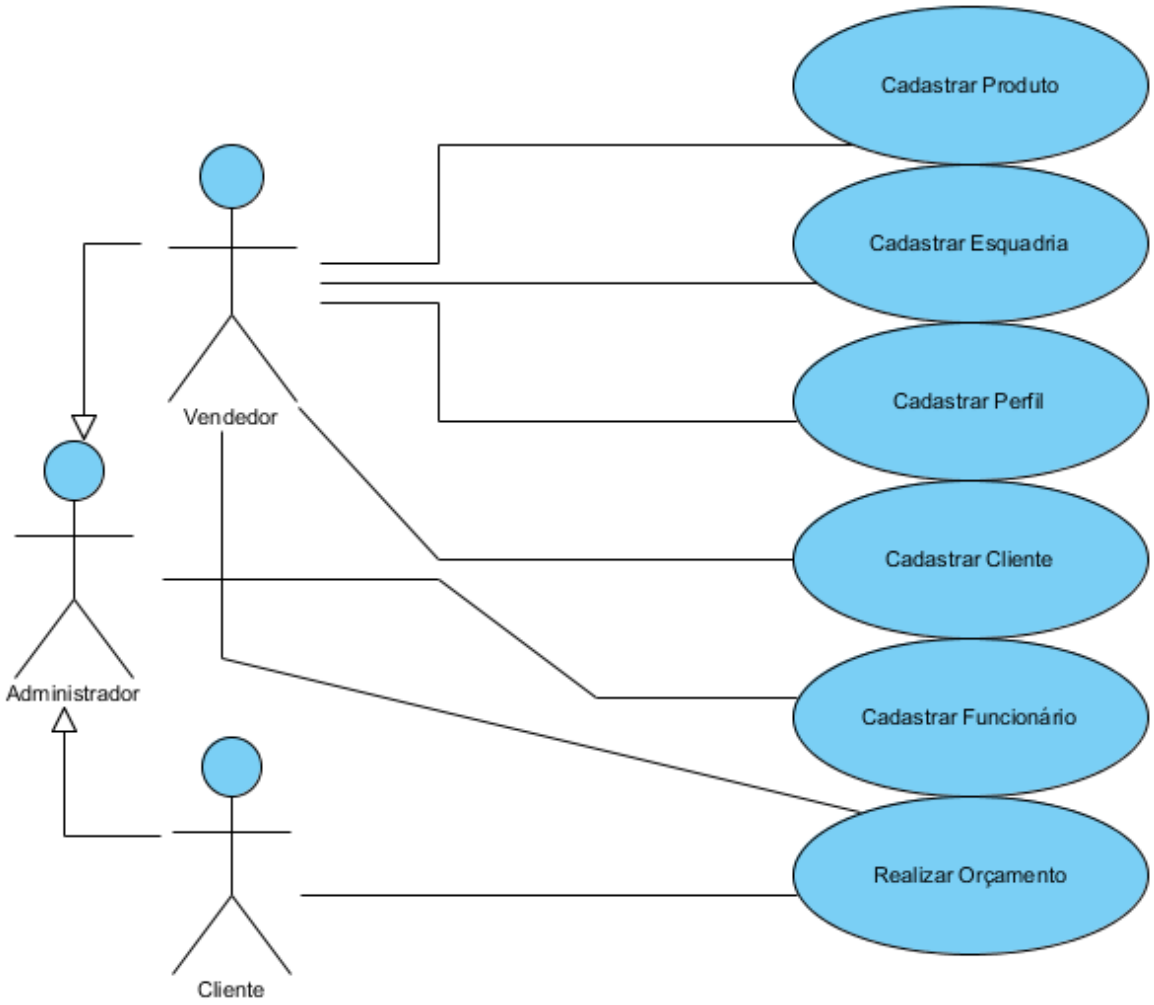

Figura 8 - Diagrama de casos de uso

O Quadro 2 apresenta a relação dos atores administrador, vendedor e cliente e os requisitos associados aos mesmos.

Ação	Descrição	Requisitos relacionados
Gerenciar funcionários	Permitir que atores adicionem, consultem, alterem ou removam funcionários.	Cadastrar funcionários Alterar funcionários Consultar funcionários Excluir funcionários
Gerenciar clientes	Permitir que atores adicionem, consultem, alterem ou removam clientes.	Cadastrar clientes Alterar clientes Consultar clientes Excluir clientes
Gerenciar perfis	Permitir que atores adicionem, consultem, alterem ou removam perfis.	Cadastrar perfis Alterar perfis Consultar perfis Excluir perfis
Gerenciar esquadrias	Permitir que atores adicionem, consultem, alterem ou removam esquadrias.	Cadastrar esquadrias Alterar esquadrias Consultar esquadrias Excluir esquadrias

Gerenciar produto	Permitir que atores adicionem, consultem, alterem ou removam produtos.	Cadastrar produto Alterar produto Consultar produto Excluir produto
Gerenciar orçamento	Permitir que atores adicionem, consultem, alterem ou removam orçamentos.	Cadastrar orçamento Alterar orçamento Consultar orçamento Excluir orçamento

Quadro 1 - Atores e requisitos

O Quadro 2 descreve o requisito funcional “Cadastrar Usuário” e requisitos não funcionais.

Requisito funcional: Cadastrar usuário		
Descrição: O sistema consistirá em um *login*, validado por usuário e senha (encriptada) com uma tabela no banco de dados e armazenando os dados na sessão. Haverá dois níveis de acesso para os usuários: normal (1) e administrador (2). O usuário deve fornecer os dados: *login* e senha.		
Requisitos não funcionais:		
Nome	**Descrição**	**Categoria**
NF 1.1 Acesso ao sistema	Cada usuário poderá fazer as alterações conforme o seu nível de acesso.	Segurança
NF 1.2 Campos obrigatórios	Os campos *login*, senha e nível, são obrigatórios para salvar o cadastro.	

Quadro 2 - Requisito cadastrar usuário

O Quadro 3 descreve o requisito funcional “Cadastrar Funcionário” e os requisitos não funcionais relacionados.

Requisito funcional: Cadastrar funcionário		
Descrição: O administrador do sistema deve cadastrar os funcionários. Funcionários são as pessoas que usarão o sistema com algumas limitações. O funcionário deve fornecer os dados: Nome, CPF, *E-mail*, Celular, Rua, Número, Bairro, CEP, Localidade.		
Requisitos não funcionais:		
Nome	**Descrição**	**Categoria**
NF 2.1 Acesso ao sistema	Cada funcionário deverá usar obrigatoriamente a sua própria “conta” com a respectiva senha, para o administrador ter um controle maior das vendas.	Segurança
NF 2.2 Campos obrigatórios	Os campos Nome, CPF, Celular e Endereço, são obrigatórios para salvar o cadastro. Os outros dados podem ser complementados em outro momento, porém, são igualmente necessários.	
NF 2.3	Os campos CPF e Celular devem apresentar	

Máscaras em campos	máscaras para entrada de dados.	

Quadro 3 - Requisito cadastrar funcionário

O Quadro 4 apresenta a descrição do requisito funcional “Cadastrar cliente” e os requisitos não funcionais relacionados.

Requisito funcional: Cadastrar cliente		
Descrição: O funcionário do setor de vendas deve cadastrar o cliente antes de fazer o orçamento. O cliente informa o seu Nome, CPF/CNPJ, *E-mail*, Telefone, Celular, Endereço, Bairro, CEP e Localidade.		
Requisitos não funcionais:		
Nome	**Descrição**	**Categoria**
NF 3.1 Pessoa física ou jurídica	O funcionário deve escolher se o cliente é pessoa física ou jurídica.	Segurança
NF 3.2 Campos obrigatórios	Os campos Nome, Telefone e Endereço, são obrigatórios para salvar o cadastro. Se for cliente pessoa jurídica o campo obrigatório não será CPF e sim Inscrição Estadual.	
NF 3.3 Nome cliente	O sistema poderá avisar se já existe um cadastro com o mesmo nome. Se o administrador desejar, o sistema bloqueará o cadastro.	
NF 3.4 Máscaras de campos	Nos campos CPF e telefone, o sistema apresentará as máscaras para cada campo.	

Quadro 4 - Requisito cadastrar cliente

O Quadro 5 apresenta a descrição do requisito funcional “Cadastrar perfil” e os requisitos não funcionais relacionados.

Requisito funcional: Cadastrar perfil		
Descrição: Os funcionários do setor de vendas devem cadastrar os perfis na empresa. Os dados serão: Descrição, Peso por Metro.		
Requisitos não funcionais:		
Nome	**Descrição**	**Categoria**
NF 4.1 Código perfil	O código do perfil deve ser gerado automaticamente.	Segurança
NF 4.2 Campos obrigatórios	Todos os campos são obrigatórios para salvar o cadastro.	

Quadro 5 - Requisito cadastrar perfil

O Quadro 6 descreve o requisito funcional “Cadastrar esquadria” e os requisitos não funcionais relacionados.

Requisito funcional: Cadastrar esquadria		
Descrição: Os funcionários devem cadastrar as esquadrias na empresa. Os dados serão: Descrição, Colocação e Perfil.		
Requisitos não funcionais:		
Nome	**Descrição**	**Categoria**
NF 5.1 Código esquadria	O código deve ser gerado automaticamente.	Segurança
NF 5.2 Campos obrigatórios	Os campos Descrição, Colocação e Perfil, são obrigatórios para salvar o cadastro.	

Quadro 6 - Requisito cadastrar esquadria

O Quadro 7 descreve o requisito funcional “Cadastrar orçamento” e os requisitos não funcionais relacionados.

Requisito funcional: Cadastrar orçamento		
Descrição: O(s) funcionário(s) ou o cliente devem cadastrar um orçamento antes de fechar a compra. A tela de orçamento será da seguinte forma: na parte superior haverá o campo para informar os dados do cliente; abaixo haverá um campo para selecionar os itens que serão orçados; juntamente com os itens, será mostrado o preço e solicitado para informar a quantidade de cada item que será orçado.		
Requisitos não funcionais:		
Nome	**Descrição**	**Categoria**
NF 6.1 Código orçamento	O código do orçamento deve ser gerado automaticamente.	Segurança
NF 6.2 Apresentação preço	Ao selecionar o item, o preço será mostrado automaticamente. Será possível alterar manualmente o preço, se necessário, apenas pelo funcionário.	
NF 6.3 Busca em cadastro	O funcionário poderá buscar o cadastro do cliente pelo nome, mas mesmo assim terá a opção de colocar diretamente o código do cliente que retornará automaticamente o seu nome ao lado.	

Quadro 7 - Requisito cadastrar orçamento

O Quadro 8 apresenta a descrição do requisito funcional “Cadastrar Produto” e os requisitos não funcionais relacionados.

Requisito funcional: Cadastrar Produto

Descrição: Os funcionários do setor de vendas podem cadastrar os produtos na empresa. Os dados serão: Esquadria e Valor.		
Requisitos não funcionais:		
Nome	**Descrição**	**Categoria**
NF 7.1 Código produto	O código do produto deve ser gerado automaticamente.	Segurança
NF 7.2 Campos obrigatórios	Todos os campos são obrigatórios para salvar o cadastro.	

Quadro 8 - Requisito cadastrar produto

A Figura 9 apresenta o diagrama de classes definido para o sistema.

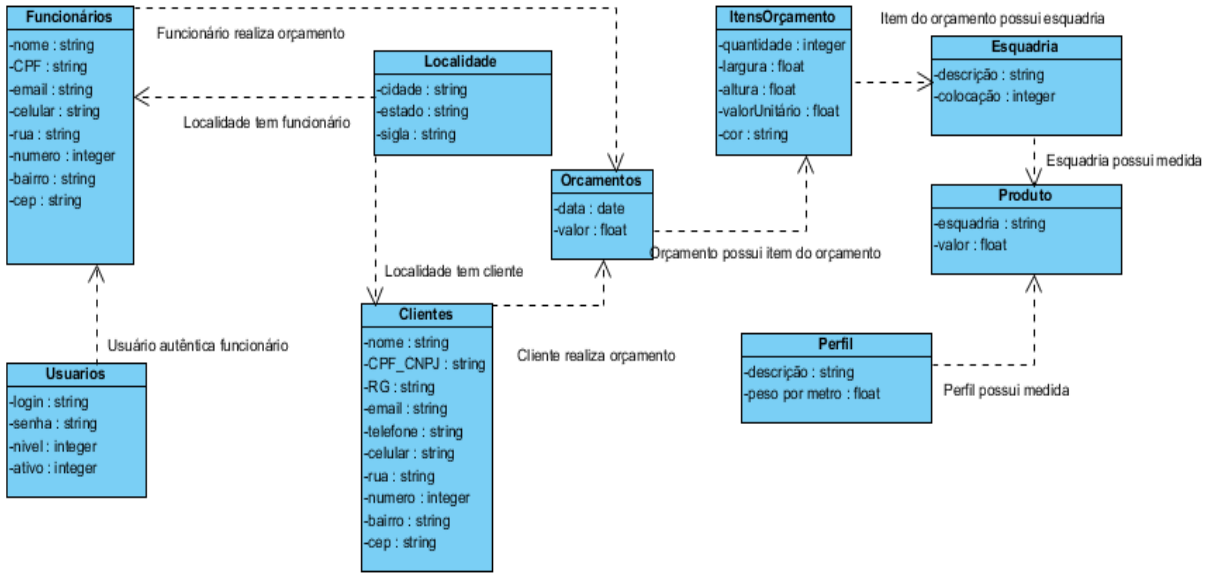

Figura 9 - Diagrama de classes

Os Quadros 9 a 17 apresentam as descrições da classe apresentada na Figura 9.

No Quadro 9 é apresentada a descrição da classe funcionário.

Identificação: Funcionário **Descrição:** Nesta classe os dados do funcionário serão requisitados, para ser gerado um código no sistema e então o cadastro é efetuado. **Atributos:** Nome (String): Nome do funcionário; CPF (String): CPF do funcionário; Email (String): Email do funcionário; Celular (String): Celular do funcionário; Endereço (String): Endereço do funcionário (Rua e número); Bairro (String): Bairro do funcionário; CEP (String): CEP do funcionário; Localidade (String): Localidade do funcionário.

Métodos:
Cadastrar
Alterar
Consultar
Excluir

Quadro 9 - Descrição da classe funcionário

No Quadro 10 é apresentada a descrição da classe cliente.

Identificação: Cliente
Descrição: Nesta classe os dados do cliente serão requisitados é gerado um código e então o cadastro é efetuado.
Atributos:
Nome (String): Nome do cliente;
CPF/CNPJ (String): CPF ou CNPJ do cliente;
Email (String): Email do funcionário;
Telefone (String): Telefone do cliente;
Celular (String): Celular do cliente;
Endereço (String): Endereço do cliente (Rua e número);
Bairro (String): Bairro do cliente;
CEP (String): CEP do cliente;
Localidade (String): Localidade do funcionário.
Métodos:
Cadastrar
Alterar
Consultar
Excluir

Quadro 10 - Classe cliente

No Quadro 11 apresenta a descrição da classe perfil.

Identificação: perfil
Descrição: Nesta classe os dados do perfil serão requisitados e então o cadastro é efetuado.
Atributos:
Descrição (String): Especificação do perfil;
Peso por metro (Float): Peso por metro de cada perfil.
Métodos:
Cadastrar
Alterar
Consultar
Excluir

Quadro 11 - Classe perfil

A descrição da classe esquadria é apresentada no Quadro 12.

Identificação: Esquadria
Descrição: Nesta classe os dados da esquadria serão requisitados é gerado um código e então o cadastro é efetuado.
Atributos:
Descrição (String): Descrição da esquadria;
Colocação (String): Colocação da esquadria;

Perfil (integer): Perfil da esquadria.
Métodos:
Cadastrar
Alterar
Consultar
Excluir

Quadro 12 - Classe esquadria

No Quadro 13 está a descrição da classe orçamento.

Identificação: Orçamento
Descrição: Nesta classe os dados do orçamento realizados pelo cliente ou funcionário serão requisitados, e então o cadastro é efetuado.
Atributos:
Data (date): Data de cadastro do orçamento;
Confirmado (Booleano): Confirmação do orçamento.
Métodos:
Cadastrar
Alterar
Consultar
Excluir

Quadro 1 - Classe orçamento

No Quadro 14 está a descrição da classe itens de orçamento.

Identificação: ItensOrçamento
Descrição: Nesta classe ocorre o cadastramento do pedido com os dados repassados pelo cliente.
Atributos:
Esquadria (float): descrição da esquadria;
Quantidade (integer): quantidade solicitada pelo cliente;
Largura (float): largura da peça;
Altura (float): altura da peça;
ValorUnitario (float): valor de cada produto;
Cor (Booleano): seleção de uma cor.
Métodos:
Cadastrar
Alterar
Consultar
Excluir

Quadro 2- Classe ItensOrcamento

No Quadro 15 está a descrição da classe usuário.

Identificação: Usuário
Descrição: Nesta classe os dados do Usuário serão solicitados e o cadastro é efetuado.
Atributos:
Login (String): Login do Usuário cadastrado;
Senha (String): Senha do Usuário cadastrado;
Nível (Integer): Nível do Usuário cadastrado;
Ativo (String): Ativo do Usuário cadastrado.

Métodos:
Cadastrar
Alterar
Consultar
Excluir

Quadro 3 - Classe usuário

No Quadro 16 está a descrição da classe produto.

Identificação: Produto
Descrição: Nesta classe ocorre o cadastramento do produto.
Atributos:
Esquadria (String): especificação da unidade de esquadria;
Valor (float): valor da esquadria.
Métodos:
Cadastrar
Alterar
Consultar
Excluir

Quadro 4- Classe Produto

No Quadro 17 está a descrição da classe localidade.

Identificação: Localidade
Descrição: Nesta classe ocorre o cadastramento das localidades (cidade, estado e sigla).
Atributos:
Cidade (String): descrição da cidade;
Estado (String): descrição do estado;
Sigla (String): descrição da sigla.
Métodos:
Cadastrar
Alterar
Consultar
Excluir
Observações:

Quadro 5 - Classe Localidade

O diagrama de entidades e relacionamentos do banco de dados do sistema é apresentado na Figura 10.

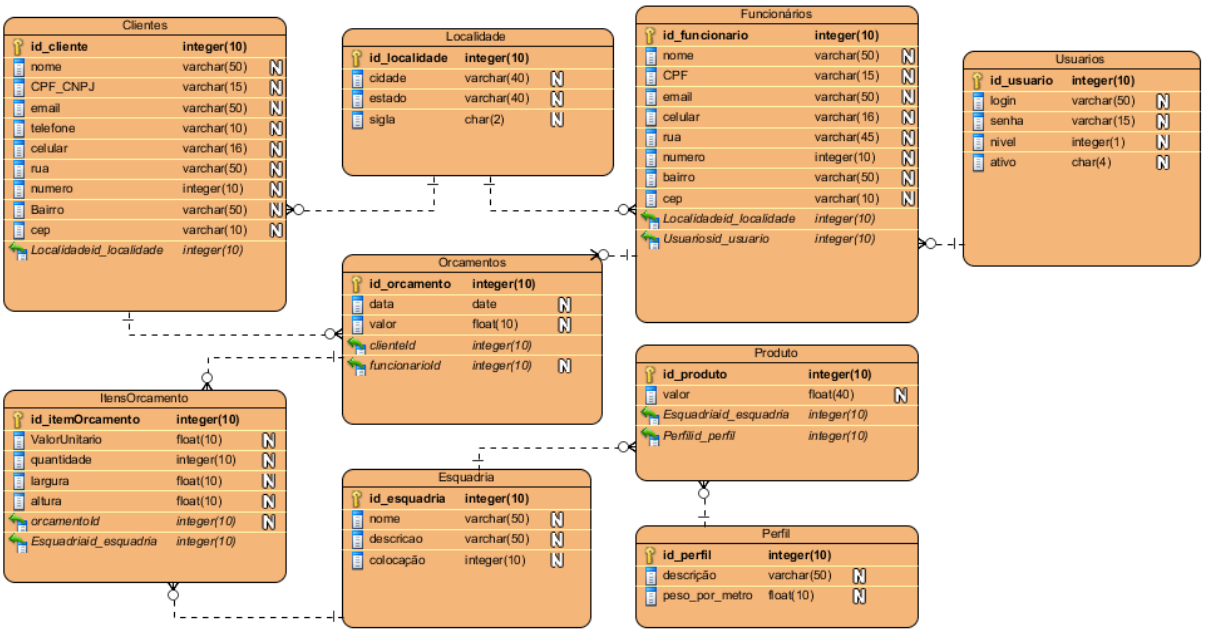

Figura 10 - Diagrama de entidades e relacionamentos

Nos Quadros 18 a 26 apresentam a descrição das tabelas do diagrama de entidade e relacionamento constante na Figura 9.

No Quadro 18 está a descrição da tabela para o cadastro de Usuários.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira	Observações
ID_USUARIO	Integer	Não	Sim	Não	Auto-Denominado
Login	String	Não	Não	Não	
Senha	String	Não	Não	Não	
Nivel	String	Não	Não	Não	
Ativo	String	Não	Não	Não	

Quadro 6 - Tabela usuários

No Quadro 19 está a descrição da tabela para o cadastro de Funcionários.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira	Observações
ID_FUNCIONARIO	Integer	Não	Sim	Não	Auto-Denominado
Nome	String	Não	Não	Não	
CPF	String	Não	Não	Não	
Email	String	Não	Não	Não	
Celular	String	Sim	Não	Não	
Rua	String	Não	Não	Não	
Numero	String	Não	Não	Não	
Bairro	String	Não	Não	Não	
CEP	String	Não	Não	Não	
ID_LOCALIDADE	Integer	Não	Não	Sim	

Usuarioid_usuario	Integer	Não	Não	Sim	

Quadro 7 - Tabela funcionários

No Quadro 20 está a descrição da tabela para o cadastro de Clientes.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira	Observações
ID_CLIENTE	Integer	Não	Sim	Sim	Auto-Denominado
Nome	String	Sim	Não	Não	
CPF_CNPJ	String	Não	Não	Não	
Email	String	Sim	Não	Não	
Telefone	String	Não	Não	Não	
Celular	String	Não	Não	Não	
Rua	String	Não	Não	Não	
Número	String	Não	Não	Não	
Bairro	String	Não	Não	Não	
Telefone	String	Sim	Não	Não	
CEP	String	Não	Não	Não	
ID_LOCALIDADE	Integer	Não	Não	Sim	

Quadro 20 - Tabela clientes

No Quadro 21 está a descrição da tabela para o cadastro de Localidade.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira	Observações
ID_LOCALIDADE	Integer	Não	Sim	Sim	Auto-Denominado
Cidade	String	Sim	Não	Não	
Estado	String	Não	Não	Não	
Sigla	String	Não	Não	Não	

Quadro 8 - Tabela localidade

No Quadro 22 está a descrição da tabela para o cadastro de Perfil.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira	Observações
ID_PERFIL	Integer	Não	Sim	Não	Auto-Denominado
Descrição	String	Não	Não	Não	
Peso_por_metro	float	Não	Não	Não	

Quadro 9 - Tabela Perfil

No Quadro 23 está a descrição da tabela para o cadastro de Esquadria.

Campo	Tipo	Nulo	Chave Primária	Chave Estrangeira	Observações
ID_ESQUADRIA	Integer	Não	Sim	Não	Auto-Denominado

Descrição	String	Não	Não	Não	
Colocação	char	Não	Não	Não	
ID_Perfil	Integer	Não	Não	Sim	

Quadro 10 - Tabela Esquadria

No Quadro 24 está a descrição da tabela para o cadastro de Produto.

Campo	**Tipo**	**Nulo**	**Chave Primária**	**Chave Estrangeira**	**Observações**
ID_PRODUTO	Integer	Não	Sim	Sim	Auto-Denominado
Valor	float	Não	Não	Não	
ID_ESQUADRIA	Integer	Não	Não	Sim	

Quadro 11 - Tabela Produto

No Quadro 25 está a descrição da tabela para o cadastro de Itens do Orçamento.

Campo	**Tipo**	**Nulo**	**Chave Primária**	**Chave Estrangeira**	**Observações**
ID_ITEMORCAMENTO	Integer	Não	Sim	Não	Auto-Denominado
EsquadriaId	Integer	Não	Não	Sim	
OrçamentoId	Integer	Não	Não	Sim	
ValorUnitário	float	Não	Não	Não	
Quantidade	Integer	Não	Não	Não	
Largura	float	Não	Não	Não	
Altura	float	Não	Não	Não	
Cor	Char	Não	Não	Não	

Quadro 12 - Tabela ItensOrçamento

No Quadro 26 está a descrição da tabela para o cadastro do Orçamento.

Campo	**Tipo**	**Nulo**	**Chave Primária**	**Chave Estrangeira**	**Observações**
ID_ORCAMENTO	Integer	Não	Sim	Não	Auto-Denominado
Data_Orcamento	Date	Não	Não	Não	
Confirmado	Integer	Não	Não	Não	
ID_FUNCIONARIO	Integer	Não	Não	Sim	
ID_CLIENTE	Integer	Não	Não	Sim	

Quadro 13 - Tabela Orçamento

4.3 APRESENTAÇÃO DO SISTEMA

Na Figura 11, o conteúdo da página está dentro da pasta “site”, sendo esta desenvolvida em HTML, CSS e JavaScript.

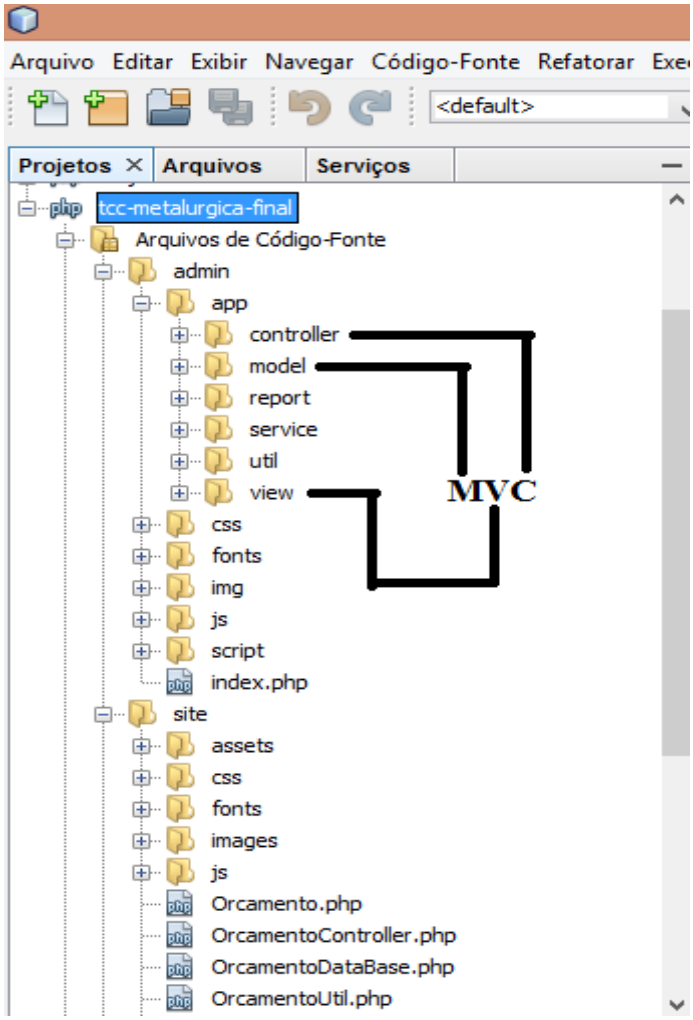

Figura 6 - Pastas e arquivos do sistema

A estrutura do projeto ficou dividida da seguinte forma: as pastas, *assets* e *images*, são responsáveis pelo armazenamento das figuras e imagens do projeto; a pasta css, possui os arquivos que estilizam as páginas, destacando-se o Bootstrap que além de estilizar é responsivo, ou seja, funciona em vários sistemas operacionais, e em diversos tamanhos de telas, tablets, celulares e outros; o arquivo owl.carousel, que faz as transições das imagens e o animate.css que realiza as animações.

Na pasta *fonts*, encontra-se os *glycons* do Bootstrap com as imagens que estão nos botões.

A pasta js contém os códigos em JavaScript responsáveis por realizar a validação de dados e o controle sobre as janelas e o documento. Ainda nessa pasta o Bootstrap que possui arquivos css e js; e o arquivo orcamento.js que responde pela execução do orçamento.

Na pasta principal do projeto (site) existem vários arquivos index. Cada um deles é responsável por uma parte da página desenvolvida, exceto os arquivos para a realização do orçamento. O Orcamento.php comunica-se com o banco de dados para a inserção, atualização e seleção dos dados. OrcamentoController comunica-se com o JavaScript. OrcamentoDataBase faz a conexão com o banco de dados e o OrcamentoUtil possui algumas funções como, por exemplo, formatar a data do formato SQL para o formato brasileiro.

A Figura 12 apresenta a tela inicial (*homepage*) da metalúrgica. No canto superior direito é apresentado o menu que permite ao cliente deslizar a barra de rolagem para conhecer o site ou selecionar o item desejado. E do lado esquerdo está o logotipo da empresa. No centro da página são disponibilizadas as categorias entre elas o item orçamento. No rodapé da página encontram-se o endereço, contato, além de ícones com *links* para as redes sociais.

Figura 7 - Tela inicial da metalúrgica

Na Figura 13 está a página com os detalhes do orçamento disponível ao usuário. No formulário para realizar o orçamento são solicitados os dados: descrição

do produto, quantidade, altura, largura e cor. O valor total calculado automaticamente. Em destaque na cor verde, o botão “Adicionar ao orçamento”. Ao clicar nesse botão, uma tabela apresenta os produtos selecionados pelo cliente e é permitido que seja feita a alteração nas descrições de cada produto, alterando, assim, o valor final do produto modificado. É possível ainda, a exclusão de um ou mais produtos. Além da possibilidade de adicionar mais itens no orçamento. Ao clicar no botão *Enviar orçamento* é apresentado um relatório.

Figura 8 - Tela de orçamento da metalúrgica

Na Figura 14 está o relatório de orçamento, com a quantidade de itens e o valor total, possibilitando ao cliente fazer o *download* ou imprimi-lo.

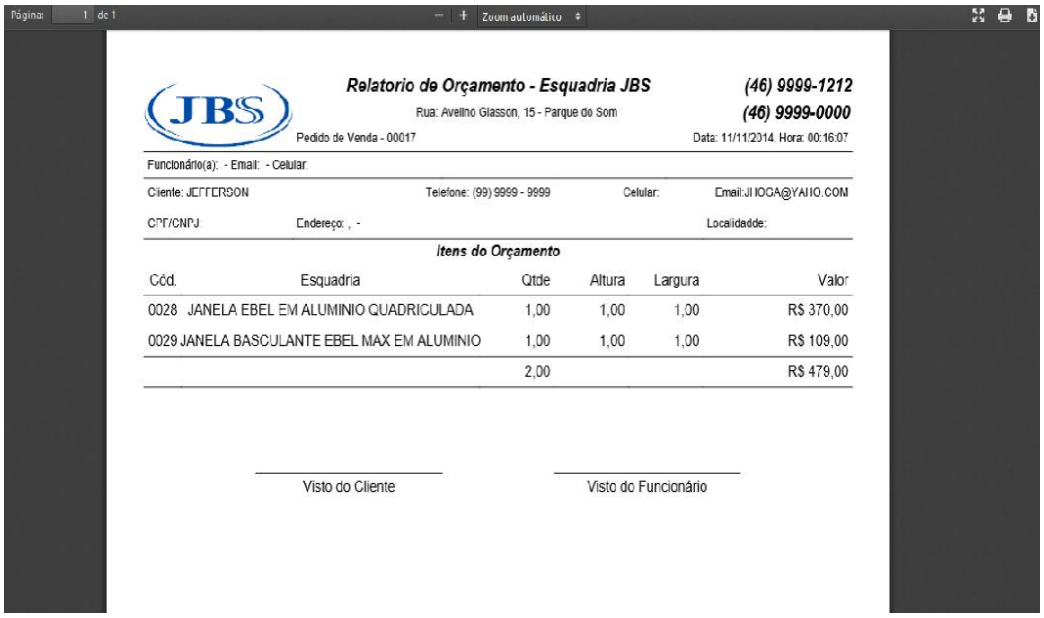

Figura 9 - Tela de relatório do orçamento da esquadria

A Figura 15 mostra a tela de acesso ao sistema. Nessa tela o funcionário preencherá os campos usuário e senha que são obrigatórios. O cadastro do usuário precisa ser validado pelo administrador para que o acesso ao sistema possa ser feito por meio de níveis de acesso.

Figura 10 - Tela de acesso ao sistema administrativo

Se o usuário cadastrado já foi ativado pelo administrador, após logar-se ele será encaminhado ao painel administrativo. Caso contrário, conforme mostra a figura

16, uma mensagem será mostrada em uma janela modal superposta a página para o usuário, alertando-o que o seu usuário ou senha ainda são inválidos.

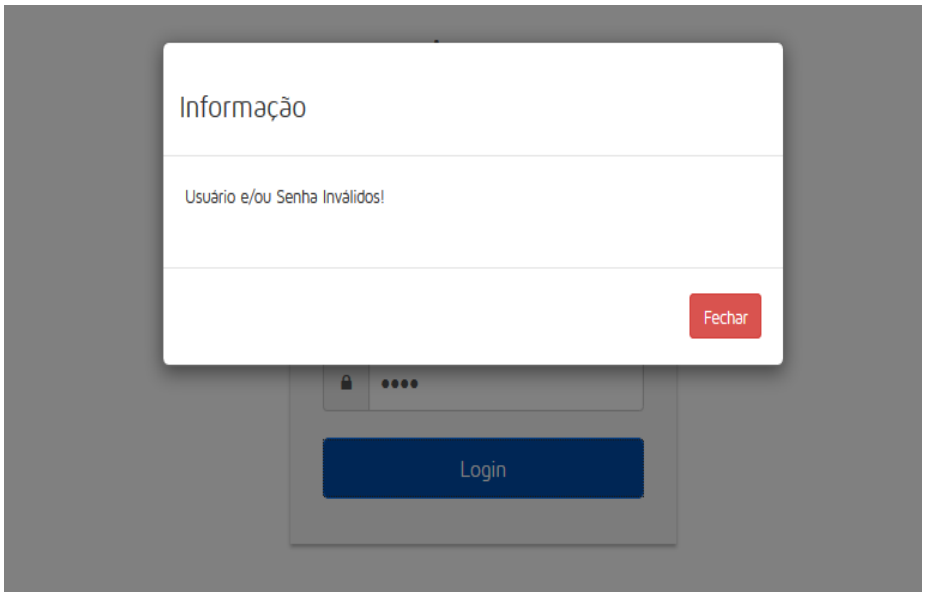

Figura 11 - Tela de alerta de usuário bloqueado

Se o usuário que efetuou o *login* é do tipo Administrador, todas as funcionalidades do sistema são disponibilizadas. Sendo o menu do painel administrativo alterado, dependendo do nível de acesso do usuário, como apresentado na Figuras 17, 18 e 19 respectivamente.

Figura 12 - Tela para o usuário administrador

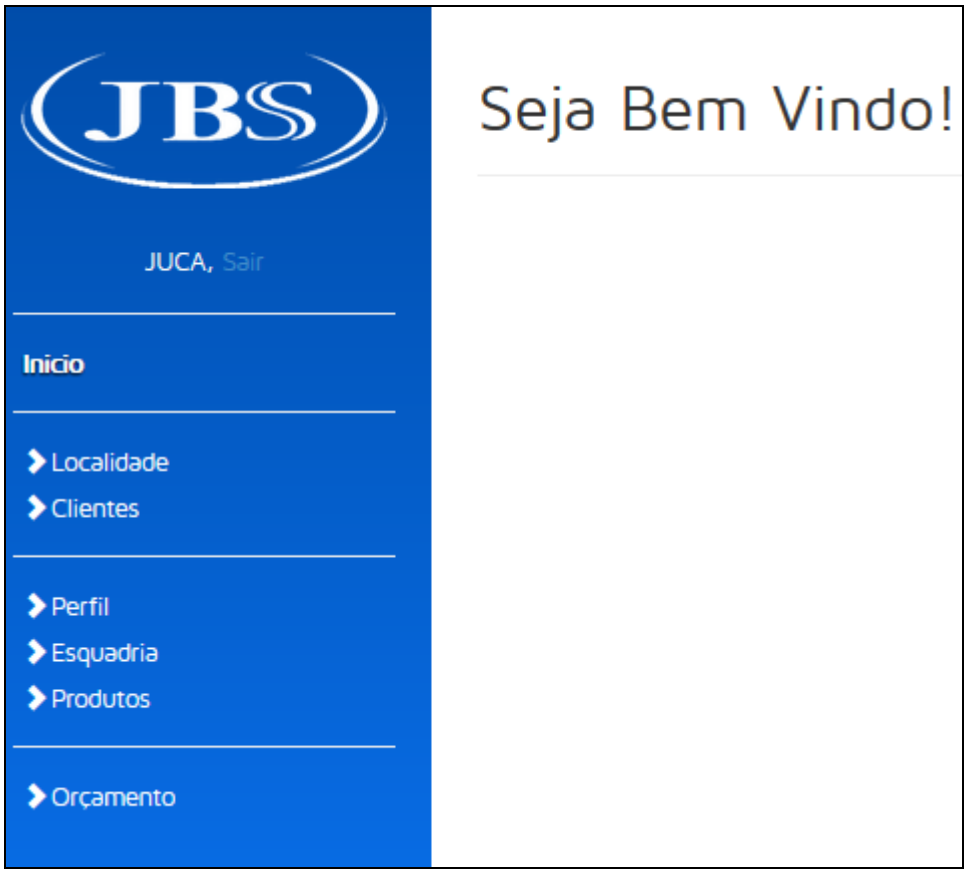

Figura 13 - Tela para o usuário gerente

Figura 14 - Tela para o usuário vendedor

A Figura 20 apresenta a tela de cadastro de produtos da metalúrgica, com dois botões, *Salvar*, o qual vai adicionar o produto no banco de dados e o *Novo*, que ao ser clicado limpará os campos.

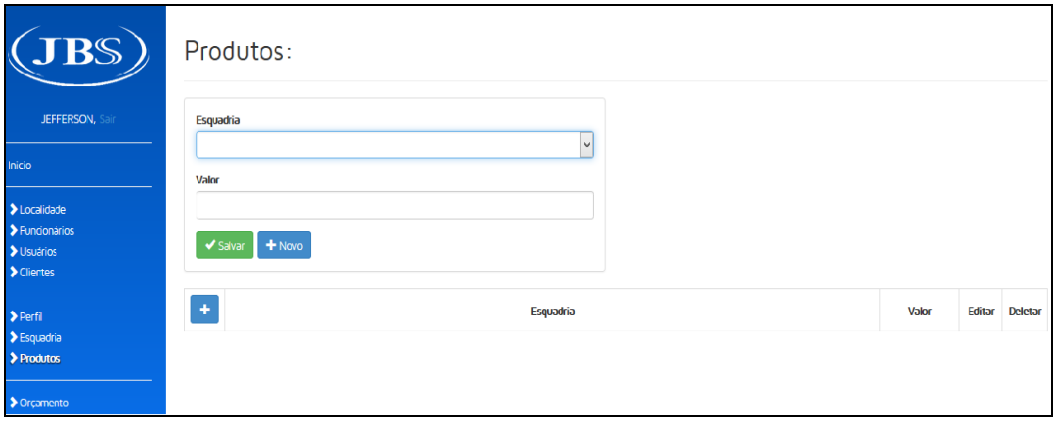

Figura 20 - Cadastrar produtos

Estando o produto devidamente cadastrado, este poderá ser editado quando houver necessidade, conforme mostra a Figura 21.

Figura 15 - Editar produtos

A Figura 22 apresenta a listagem de produtos cadastrados. Por meio dela, pode-se editar ou excluir produtos.

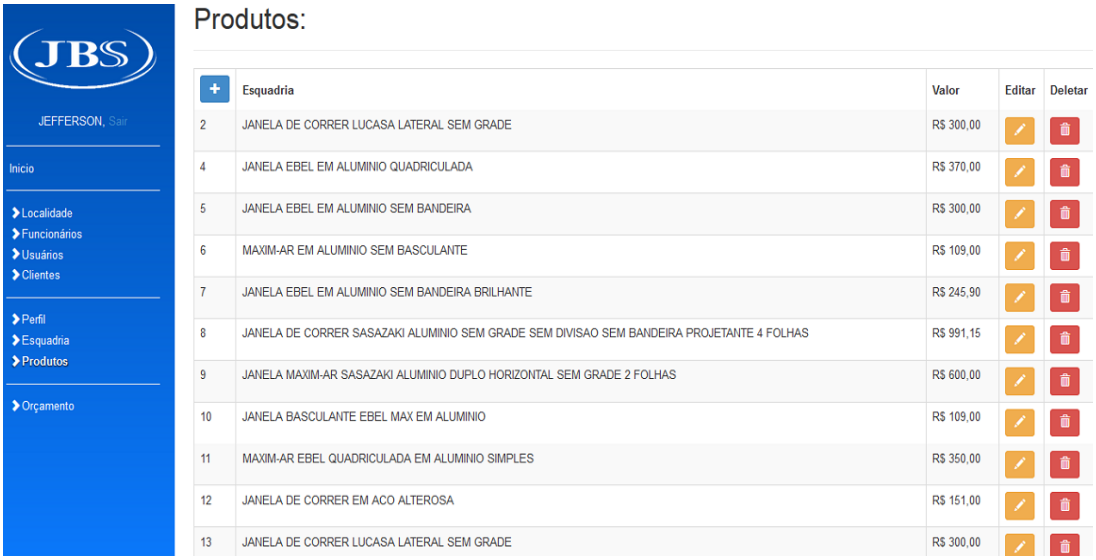

Figura 16 - Lista de produtos

A Figura 23 mostra um botão para excluir um ou mais produtos. Antes de confirmar a exclusão é apresentado um alerta em modal perguntando se deseja excluir o registro. Se confirmada a operação, o produto será excluído definitivamente do banco de dados.

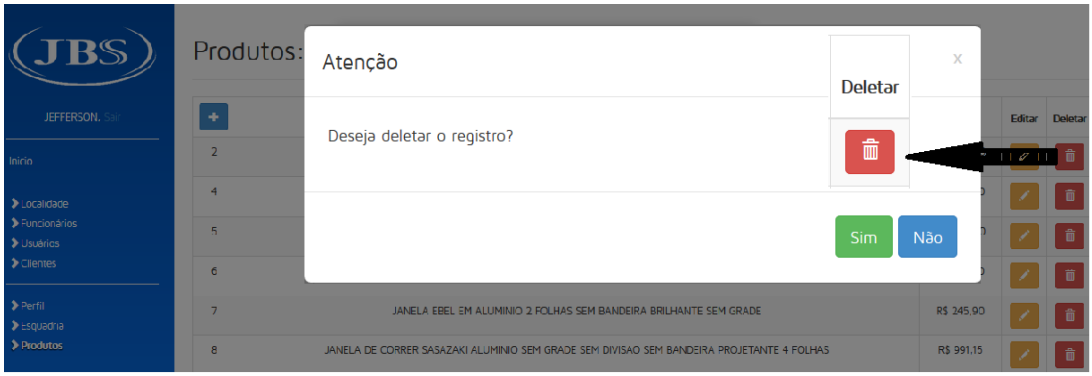

Figura 17 - Confirmar exclusão de registro

Semelhante ao cadastro de produtos, a Figura 24 apresenta a tela de cadastro de orçamentos. Ao clicar no botão “salvar”, o sistema mostra os campos das tabelas orçamento e itens do orçamento, necessários para a realização do orçamento.

Figura 18 - Cadastrar orçamento

Ainda na Figura 24 existe o botão adicionar, possibilitando ao usuário uma pré-visualização dos itens do orçamento, antes de serem inseridos no banco de dados. E ao clicar no referido botão este apresenta as informações em uma tabela, conforme a Figura 25. É disponibilizado, ainda, o botão em vermelho, excluir, caso necessite excluir um ou mais registros.

Figura 19 - Pré-visualização dos itens do orçamento

A Figura 26 além de listar a data do orçamento, funcionário e cliente cadastrados na tabela orçamentos, possui os seguintes botões, Editar, Excluir, Lupa e Relatório; Ao clicar na “lupa”, uma prévia dos itens do orçamento elaborados pelo funcionário é apresentado em uma janela modal, com os produtos e o valor total do orçamento já cadastrados no banco de dados, conforme figura 27. Por fim a Figura 28 apresenta a tela com o relatório correspondente ao orçamento.

Figura 20 - Botões Editar, Excluir, Lupa, e Relatório

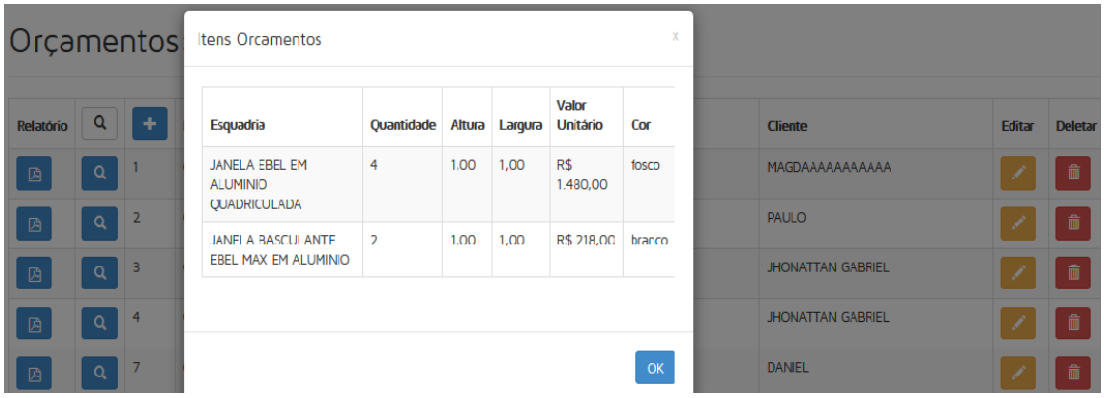

Figura 21 - Tela modal dos itens do orçamento

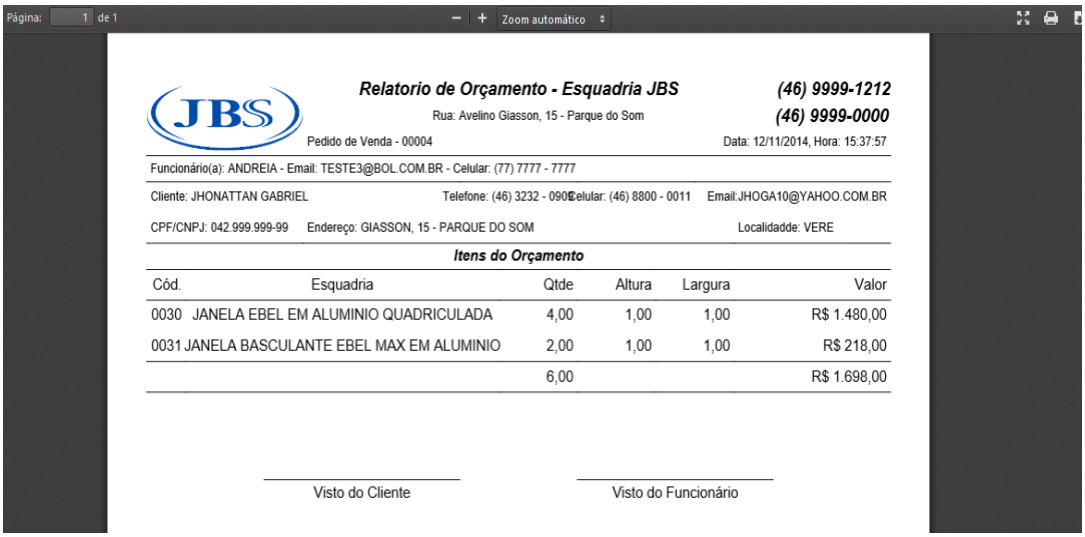

Figura 22 - Relatório do orçamento elaborado pelo funcionário

4.4 IMPLEMENTAÇÃO DO SISTEMA

Nesta seção são apresentadas partes dos principais códigos desenvolvidos na implementação do sistema.

A Listagem 1 apresenta o trecho de código responsável pela comunicação com o banco de dados no qual é realizado o *insert*, que se encontra no arquivo orcamento.php. Estando os dados preenchidos corretos, eles são gravados no banco de dados.

```
public function salvarCliente( $params ) {
        $ID_CLIENTE    = utf8_decode( 0 );
        $NOME          = utf8_decode( mb_strtoupper( $params['NOME'], 'UTF-8' ));
        $EMAIL         = utf8_decode( mb_strtoupper( $params['EMAIL'], 'UTF-8' ));
        $TELEFONE      = utf8_decode( $params['TELEFONE'] );
       $sql  = "";
        $sql .= " SELECT ID_CLIENTE ";
        $sql .= " FROM CLIENTE ";
              $sql .= " WHERE EMAIL = '$EMAIL' ";
              $retorno = $this->database->select_sql( $sql );
              if ( $retorno == false ) {
                     return (int) $this->database->execute_sql_last_id(" INSERT INTO
CLIENTE(ID_CLIENTE, NOME, EMAIL, TELEFONE)
                                   VALUES($ID_CLIENTE, '$NOME', '$EMAIL', '$TELEFONE') ");
              }
              else {
```

```
                    return (int) $retorno[0]['ID_CLIENTE'];
            }
    }
public function salvarOrcamento( $params, $ID_CLIENTE ) {
        $ID_ORCAMENTO   = utf8_decode(($params['ID_ORCAMENTO'] == '') ? 0 :
$params['ID_ORCAMENTO']);
       $DATA_ORCAMENTO  = utf8_decode( date('Y-m-d') );
       $CONFIRMADO     = utf8_decode( 0 );
       $ID_FUNCIONARIO  = 0;

       return (int) $this->database->execute_sql_last_id(" INSERT INTO
ORCAMENTO(DATA_ORCAMENTO, CONFIRMADO, ID_FUNCIONARIO, ID_CLIENTE)
            VALUES('$DATA_ORCAMENTO', $CONFIRMADO, $ID_FUNCIONARIO, $ID_CLIENTE) ");
    }

    public function salvarItensOrcamentos( $params, $id_orcamento) {
       $id_esquadria   = utf8_decode(($params['id_esquadria'] == '') ? 0 :
$params['id_esquadria']);
       $qtde           = utf8_decode($params['qtde']);
       $altura         = utf8_decode(Utils::formatCurrency($params['altura']));
       $largura        = utf8_decode(Utils::formatCurrency($params['largura']));
       $valor_unitario = utf8_decode(Utils::formatCurrency($params['valor_unitario']));
       $cor            = utf8_decode($params['cor']);
       return (int) $this->database->execute_sql(" INSERT INTO
ITEM_ORCAMENTO(ID_ESQUADRIA, QUANTIDADE, ALTURA, LARGURA,
       VALOR_UNITARIO, COR, ID_ORCAMENTO)VALUES($id_esquadria, $qtde, $altura,
$largura, $valor_unitario, $cor, $id_orcamento) ");
    }
```

Listagem 1 - Inclusão de dados no orçamento

A Listagem 2 apresenta o código utilizado no relatorioOrcamento.php que faz o *select* no banco de dados, buscando os dados do cliente e os itens do orçamento para fazer a montagem do relatório.

```
public function cabecalho() {
            $sql = " SELECT O.ID_ORCAMENTO AS CODIGO, O.DATA_ORCAMENTO AS DATA,
                        O.CONFIRMADO, F.ID_FUNCIONARIO, F.NOME,F.CELULAR,F.EMAIL,
                        E.NOME AS CLIENTE, E.TELEFONE AS
CLIENTE_TELEFONE,E.RUA,E.NUMERO,E.BAIRRO,E.CPF_CNPJ,
                        E.CELULAR AS CLIENTE_CELULAR, E.EMAIL AS CLIENTE_EMAIL,
L.CIDADE
                    FROM ORCAMENTO O
```

```
                    LEFT JOIN FUNCIONARIO F
                    ON O.ID_FUNCIONARIO = F.ID_FUNCIONARIO
                    LEFT JOIN CLIENTE E
                    ON O.ID_CLIENTE = E.ID_CLIENTE
                    LEFT JOIN LOCALIDADE L
                    ON E.ID_LOCALIDADE = L.ID_LOCALIDADE
                    WHERE O.ID_ORCAMENTO = " . $this->id;

            $r = $this->database->select_sql( $sql );
            return $r[0];
      }
      public function itensOrcamentos() {
            $sql = " SELECT ID_ITEM_ORCAMENTO AS CODIGO, QUANTIDADE, ALTURA, LARGURA,
VALOR_UNITARIO, COR, E.DESCRICAO
                    FROM ITEM_ORCAMENTO I
                    INNER JOIN ESQUADRIA E
                    ON I.ID_ESQUADRIA = E.ID_ESQUADRIA
                    WHERE I.ID_ORCAMENTO = " . $this->id;
            return $this->database->select_sql( $sql );
      }
}
```

Listagem 2 - Montagem do relatório de orçamento

O código utilizado na Listagem 3 realiza a comunicação com o banco de dados e estando o usuário cadastrado e ativo ele é redirecionado para o painel administrativo.

```
public function logar( $params=null ) {
        $usuario = strtoupper(utf8_decode($params["usuario"]));
        $senha = strtoupper(utf8_decode($params["senha"]));

        $sql  = "";
        $sql .= " SELECT *
            FROM USUARIOS
            WHERE LOGIN = '$usuario'
            AND SENHA = '$senha'
            AND ATIVO = 1 ";
        $retorno = $this->database->select_sql( $sql );
        if ( $retorno != false ) {
            Session::create( $retorno[0] );
            return true;
        }
```

```
        return false;
    }
}
```

Listagem 3 - Logar no Sistema

O trecho de código utilizado na Listagem 4 é executado quando o usuário selecionar o botão *login*. Dessa forma, se os dados do cadastro não estiverem corretos, uma nova janela *modal* será aberta e através da função alert() é mostrada a descrição que usuário ou senha estão inválidos.

```
 logar : function() {
             var params = {
                    'metodo' : 'logar',
                    'formulario' : $("#form-login").serialize()
             };
             $.post('?m=controller&c=LoginController', params, function( data ) {
                    if ( typeof data === "boolean" && data == true ) {
                           window.setTimeout(function() { window.location.href =
'?m=view'; }, 1000);
                    }
                    else {
                           Alert.show('Usuário e/ou Senha Inválidos!');
                    }
             }, 'json');
       }
};
```

Listagem 4 - Código para confirmar usuário e senha no sistema

O trecho de código para a apresentação da tela do painel administrador, está na Listagem 5. Conforme o nível de acesso do usuário é liberado a lista de opções do menu.

```
<div id="menu" class="navbar-collapse collapse sidebar-navbar-collapse">
                    <ul class="nav navbar-nav">
                           <li><hr/></li>
                           <li class="active"><a id="menu-inicio"
href="javascript:void(0)" >Inicio</a></li>
```

```
                           <li><hr/></li>
                           <?php if ( (Session::get('NIVEL_ACESSO') == 0) ||
(Session::get('NIVEL_ACESSO') == 1) ) { // ouro & prata ?>
                           <li><a id="menu-localidade" href="javascript:void(0)"
><span class="glyphicon glyphicon-chevron-right"></span>Localidade</a></li>
```

```
                            <?php  if ( Session::get('NIVEL_ACESSO') == 0 ) { // ouro
?>
                            <li><a id="menu-funcionario" href="javascript:void(0)"
><span class="glyphicon glyphicon-chevron-right"></span>Funcionários</a></li>
                            <li><a id="menu-usuarios" href="javascript:void(0)" ><span
class="glyphicon glyphicon-chevron-right"></span>Usuários</a></li>
                            <?php  } ?>
                            <?php } ?>
                            <li><a id="menu-cliente" href="javascript:void(0)" ><span
class="glyphicon glyphicon-chevron-right"></span>Clientes</a></li>
                            <li><hr/></li>
                            <?php if ( (Session::get('NIVEL_ACESSO') == 0) ||
(Session::get('NIVEL_ACESSO') == 1) ) { // ouro & prata ?>
                            <li><a id="menu-perfil" href="javascript:void(0)" ><span
class="glyphicon glyphicon-chevron-right"></span>Perfil</a></li>
                            <li><a id="menu-esquadria" href="javascript:void(0)" ><span
class="glyphicon glyphicon-chevron-right"></span>Esquadria</a></li>
                            <li><a id="menu-produto" href="javascript:void(0)" ><span
class="glyphicon glyphicon-chevron-right"></span>Produtos</a></li>
                            <li><hr/></li>
                            <?php } ?>
                            <li><a id="menu-orcamento" href="javascript:void(0)" ><span
class="glyphicon glyphicon-chevron-right"></span>Orçamento</a></li>
                            </ul>
                    </div>
```

Listagem 5 - Código para tela do painel administrador

O trecho de código utilizado para cadastrar, listar, editar e excluir os produtos encontra-se na Listagem 6.

```
public function buscar( $params ) {
       $sql  = "";
       $sql .= " SELECT PRODUTO.*, ESQUADRIA.DESCRICAO AS ESQUADRIA ";
       $sql .= " FROM PRODUTO ";
       $sql .= " INNER JOIN ESQUADRIA ";
       $sql .= " ON PRODUTO.ID_ESQUADRIA = ESQUADRIA.ID_ESQUADRIA ";
       $sql .= " ORDER BY 1 ";
           $retorno = $this->database->select_sql( $sql );
              foreach ($retorno as $key => $value) {
                     $retorno[ $key ][ 'VALOR' ] =
utf8_encode(Utils::formatCurrencyBr($value['VALOR']));
```

```
            }
            return $retorno;
    }

       public function buscarEsquadria( $params ) {
        $sql  = "";
        $sql .= " SELECT * ";
        $sql .= " FROM ESQUADRIA ";

           $retorno = $this->database->select_sql( $sql );
            foreach ($retorno as $key => $value) {
                  $retorno[ $key ][ 'DESCRICAO' ] = utf8_encode( $value['DESCRICAO'] 
);
                  $retorno[ $key ][ 'COLOCACAO' ] = utf8_encode( $value['COLOCACAO'] 
);
            }
            return $retorno;
    }

       public function editar( $params ) {
            $code = utf8_decode($params['codigo']);
        $sql  = "";
        $sql .= " SELECT * FROM PRODUTO ";
        $sql .= " WHERE ID_PRODUTO = " . $code;

            $retorno = $this->database->select_sql( $sql );
            $retorno[0][ 'VALOR' ] = Utils::formatCurrencyBr($retorno[0][ 'VALOR' ]);
            return $retorno[0];
    }

    public function salvar( $params ) {
        $ID_PRODUTO     = utf8_decode(($params['ID_PRODUTO'] == '') ? 0 : 
$params['ID_PRODUTO']);
        $VALOR          = utf8_decode(Utils::formatCurrency($params[ 'VALOR' ]));
        $ID_ESQUADRIA   = utf8_decode($params['ID_ESQUADRIA']);
            if (  $params['ID_PRODUTO'] > 0 ) {
                  return (int) $this->database->execute_sql(" UPDATE PRODUTO SET 
VALOR='$VALOR', ID_ESQUADRIA='$ID_ESQUADRIA' WHERE ID_PRODUTO='$ID_PRODUTO' ");
            }
        else {
                  return (int) $this->database->execute_sql(" INSERT INTO 
PRODUTO(ID_PRODUTO, VALOR, ID_ESQUADRIA)
```

```
                            VALUES($ID_PRODUTO, $VALOR, $ID_ESQUADRIA) ");
            }
    }
       public function excluir( $params ) {
              $codigo = utf8_decode($params['codigo']);
              return $this->database->execute_sql("DELETE FROM PRODUTO WHERE ID_PRODUTO
= $codigo ");
       }
```

Listagem 6 - Cadastro, lista, edição e remoção de produtos

Nas listagens 7 e 8 são apresentados os arquivos que utilizaram o *Free Portable Document Format* (FPDF) que é uma classe PHP, o qual permite gerar arquivos PDF com PHP sem o uso da biblioteca PDFlib. Para apresentar o relatório que fica na pasta admin/app/report, estão os trechos dos códigos dos arquivos, relatorioOrcamentoPDF.php e relatorioOrcamento.php, esse faz o *select* no banco de dados e busca o cabeçalho e os itens do orçamento para formar o referido relatório, já o relatorioOrcamentoPDF.php, monta o PDF, e o index.php apresenta o documento no navegador.

```
public function cabecalho() {
              $sql = " SELECT O.ID_ORCAMENTO AS CODIGO, O.DATA_ORCAMENTO AS DATA,
                            O.CONFIRMADO, F.ID_FUNCIONARIO, F.NOME,F.CELULAR,F.EMAIL,
                            E.NOME AS CLIENTE, E.TELEFONE AS
CLIENTE_TELEFONE,E.RUA,E.NUMERO,E.BAIRRO,E.CPF_CNPJ,
                            E.CELULAR AS CLIENTE_CELULAR, E.EMAIL AS CLIENTE_EMAIL,
L.CIDADE
                     FROM ORCAMENTO O
                     LEFT JOIN FUNCIONARIO F
                     ON O.ID_FUNCIONARIO = F.ID_FUNCIONARIO
                     LEFT JOIN CLIENTE E
                     ON O.ID_CLIENTE = E.ID_CLIENTE
                     LEFT JOIN LOCALIDADE L
                     ON E.ID_LOCALIDADE = L.ID_LOCALIDADE
                     WHERE O.ID_ORCAMENTO = " . $this->id;

              $r = $this->database->select_sql( $sql );
              return $r[0];
       }

       public function itensOrcamentos() {
```

```
            $sql = " SELECT ID_ITEM_ORCAMENTO AS CODIGO, QUANTIDADE, ALTURA, LARGURA,
VALOR_UNITARIO, COR, E.DESCRICAO
                    FROM ITEM_ORCAMENTO I
                    INNER JOIN ESQUADRIA E
                    ON I.ID_ESQUADRIA = E.ID_ESQUADRIA
                    WHERE I.ID_ORCAMENTO = " . $this->id;

            return $this->database->select_sql( $sql );
      }
```

Listagem 7 - Busca dos dados para formar o relatório

```
function Header()
  {
    $this->SetFont('Arial','BI',12);
    $this->Cell(30,6, utf8_decode(' '),0,0,'L');
    $this->Cell(130,6, utf8_decode('Relatorio de Orçamento - Esquadria JBS'),0,0,'C');
    $this->Cell(30,6, utf8_decode('(46) 9999-1212'),0,1,'L');
    $this->SetFont('Arial','',8);
    $this->Cell(40,6, utf8_decode(''),0,0,'C');
       $this->Cell(120,6, utf8_decode("Rua: Avelino Giasson, 15 - Parque do
Som"),0,0,'C');
       $this->SetFont('Arial','BI',12);
       $this->Cell(30,6, utf8_decode('(46) 9999-0000'),0,1,'C');
       $this->SetFont('Arial','',8);
    $this->Cell(40,6, utf8_decode(''),0,0,'C');
       $this->Cell(108,6, utf8_decode("Pedido de Venda - " . Utils::addZeros($this->id,
5)),0,0,'L');
       $this->Cell(21,6, utf8_decode("Data: ".date('d/m/Y'). ","),0,0,'C');
       $this->Cell(21,6, utf8_decode("Hora: ".date('H:i:s')),0,1,'R');
       // $this->Line(10, 29, 200, 29);
       $cab = $this->rel->cabecalho();
       $this->SetFont('Arial','',8);
    $this->Cell(0,6, utf8_decode('Funcionário(a): ' . $cab['NOME'] . ' - Email: ' .
$cab['EMAIL'] .' - Celular: '. $cab['CELULAR']),'T',1,'L');
    $this->Cell(80,7, 'Cliente: ' . $cab['CLIENTE'],'T',0,'L');
    $this->Cell(30,7, 'Telefone: ' . $cab['CLIENTE_TELEFONE'], 'T',0,'R');
    $this->Cell(30,7, 'Celular: ' . $cab['CLIENTE_CELULAR'], 'T',0,'R');
    $this->Cell(50,7, 'Email:'. $cab['CLIENTE_EMAIL'], 'T',1,'R');
    $this->Cell(40,7, 'CPF/CNPJ: ' . $cab['CPF_CNPJ'],'B',0,'L');
```

```
    $this->Cell(110,7, utf8_decode('Endereço: ') . $cab['RUA'].', '.$cab['NUMERO'].' -
'.$cab['BAIRRO'],'B',0,'L');
    $this->Cell(40,7, 'Localidadde: ' . $cab['CIDADE'], 'B',1,'L');
    $this->SetTextColor(0,0,0);
    $this->SetFillColor(255,255,255);
    $this->SetFont('Arial','BI',10);
    $this->Cell(0,6, utf8_decode('Itens do Orçamento'),0,1,'C');
    $this->SetFont('Arial','',10);
    $this->Cell(10,7, utf8_decode('Cód.'),'BT',0,'R');
    $this->Cell(80,7, 'Esquadria','BT',0,'C');
    $this->Cell(20,7, 'Qtde','BT',0,'R');
    $this->Cell(20,7, 'Altura','BT',0,'R');
    $this->Cell(20,7, 'Largura','BT',0,'R');
    $this->Cell(40,7, 'Valor','BT',1,'R');
    $this->Image(FPDF . DS . 'logo.png', 10, 12, 40);
  }
  function Footer()
  {
    $this->SetY(-15);
    $this->SetFont('Arial','I',8);
    $this->Cell(70,10,'Impresso em: ' . date('d/m/Y H:i:s'),0,0,'L');
    $this->Cell(50,10,$this->PageNo().' / {nb}',0,0,'C');
    $this->Cell(70,10,'Metalurgica vs 0.0.1',0,1,'R');
  }
  public function show()
  {
       $qtd= 0;
       $vlr= 0;
    foreach($this->rel->itensOrcamentos() as $key => $values) {
              $this->Cell(10,7, Utils::addZeros($values['CODIGO'], 4),'',0,'R');
              $this->Cell(80,7, $values['DESCRICAO'],'',0,'C');
              $this->Cell(20,7,
Utils::formatCurrencyBr($values['QUANTIDADE']),'',0,'R');
              $this->Cell(20,7,  Utils::formatCurrencyBr($values['ALTURA']),'',0,'R');
              $this->Cell(20,7,  Utils::formatCurrencyBr($values['LARGURA']),'',0,'R');
              $this->Cell(40,7, 'R$ ' .
Utils::formatCurrencyBr($values['VALOR_UNITARIO']),'',1,'R');

              $qtd += $values['QUANTIDADE'];
              $vlr += $values['VALOR_UNITARIO'];
    }
    $this->Cell(10,7, utf8_decode(''),'BT',0,'R');
```

```
    $this->Cell(80,7, '','BT',0,'C');
    $this->Cell(20,7, Utils::formatCurrencyBr($qtd),'BT',0,'R');
    $this->Cell(20,7, '','BT',0,'R');
    $this->Cell(20,7, '','BT',0,'R');
    $this->Cell(40,7,'R$ '. Utils::formatCurrencyBr($vlr),'BT',1,'R');
    $this->Ln(20);
$this->Cell(30,6, utf8_decode(''),'',0,'C');
       $this->Cell(50,6, utf8_decode("Visto do Cliente"),'T',0,'C');
       $this->Cell(30,6, utf8_decode(''),'',0,'C');
       $this->Cell(50,6, utf8_decode("Visto do Funcionário"),'T',0,'C');
       $this->Cell(30,6, utf8_decode(''),'',1,'C');
  }
```

Listagem 8 - Montagem do PDF para o relatório de orçamento

4.5 DISCUSSÃO

O sistema foi implementado em dois módulos. Um módulo para o cliente realizar o orçamento e outro para o gerenciamento administrativo, que também tem acesso à funcionalidade de orçamento.

O que está colocado nesta seção tem como base o ponto de vista do programador. Assim, foram considerados aspectos relacionados à implementação decorrentes de uma análise informal, que tem o objetivo de apenas colocar o ponto de vista no desenvolvimento do projeto. O sistema foi implementado utilizando PHP, JavaScript, HTML e CSS. Deste modo o projeto, permaneceu com uma estrutura padrão para todo o seu desenvolvimento.

Assim, utilizando o *login* como exemplo. Para o desenvolvimento da interface visual está HTML, conforme exemplo que se encontra na pasta app/view/mn-login.php, na sequência é desenvolvido o JavaScript e colocado na pasta JS e este se comunica com o controller/login.js, que está na pasta controller. Depois de criado o *controller*, que está na pasta app/controller, ocorre a comunicação com o diretório *model* e js/loginController.php. O *model* por sua vez se comunica com o banco de dados, realizando conforme o arquivo, o *select*, *update* e *delete*, os quais estão na pasta app/model.

A estrutura e divisão das pastas dos projetos permaneceram então da seguinte forma: a app/controller, responsável por se comunicar com o JavaScript e o

model que se comunica com o banco de dados. O diretório *report* contém os arquivos utilizados no desenvolvimento do relatório de orçamento. A pasta *service* tem o arquivo que se conecta com o banco de dados, já para algumas funções úteis ao projeto como exemplo a conversão da data, arquivos que se encontram no diretório útil. Na pasta *view* estão os HTMLs. O CSS por sua vez não é específico de cada tela, existindo um geral para o projeto. Em *fonts* há as fontes do *framework* Bootstrap nele existem imagens conhecidas como *glycons*, na pasta img as demais imagens do projeto e por fim no js, arquivos JavaScript que se comunicam com o *controller* e a *view* (HTML).

CONCLUSÃO

O objetivo deste trabalho foi alcançado com o desenvolvimento de um sistema *Web* para uma indústria metalúrgica de pequeno porte. A funcionalidade principal em termos de negócio do sistema está na realização do orçamento que pode ser feita pelo cliente.

Em termos de modelagem e implementação do sistema observou-se que a ferramenta Visual Paradigm auxiliou muito na realização da análise por fornecer uma visão geral do sistema de maneira clara.

Este trabalho foi desenvolvido também para o âmbito comercial, assim, com o intuito de apresentar o sistema da empresa de esquadria de alumínio, então se pretendeu desenvolver uma interface que atende aos critérios de usabilidade para que o sistema possa ser facilmente utilizado pelos funcionários da empresa e por clientes que podem possuir pouca habilidade no uso de sistemas computacionais.

Em trabalhos futuros, poderão ser realizadas integrações com outros setores da empresa. A geração da imagem da esquadria para visualização de como a mesma vai ficar antes de sua instalação em uma residência é outra funcionalidade interessante que o sistema poderia implementar. E ainda poderá ser implementado o cálculo de orçamento para todos os tipos de esquadrias e o controle de estoque. Na versão atual do sistema não é realizado o cálculo de orçamento de portões e cercas, que não utilizam da tabela medidas como foi definida. Uma forma de otimização para melhor aproveitamento e menor perda de material, de acordo com os perfis disponíveis em estoque no momento do orçamento, é outra funcionalidade que poderá ser implementada.

REFERÊNCIAS BIBLIOGRÁFICAS

APACHE XAMPP. **Apache Xampp**. Disponível em: <https://www.apachefriends.org/download.html>. Acesso em: 10 ago. 2014.

BEZERRA, Eduardo. **Princípios de análise e projeto de sistemas com UML**. Rio de Janeiro: Elsevier, 2007.

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar. **UML Guia do Usuário**, 7ª edição**,** Rio de Janeiro: Campus, 2000.

BOOTSTRAP FRAMEWORK. **Apostila Desenvolvimento *Web* com HTML, CSS e JavaScript**. Disponível em: <http://www.caelum.com.br/apostila-html-css-javascript/bootstrap-e-formularios-html5/>. Acesso em 26 set. 2014.

BURBECK, Steve. **Applications programming in smalltalk-80: how to use model-view-controller**. Acesso em: 30 Set. 2014 Disponível em: http://st-www.cs.illinois.edu/users/smarch/st-docs/mvc.html Disponível em: <http://folk.uio.no/trygver/2003/javazone-jaoo/MVC_pattern.pdf>. Acesso em 26 set. 2014.

CARVALHO Nino. **Quando e como não utilizar as redes sociais**. Disponível em: <http://www.ninocarvalho.com.br/category/cases/>. Acesso em: 10 ago. 2014

DALFOVO, Oscar. **Desenho de um modelo de sistemas de informação**. Dissertação (Mestrado em Administração de Negócios) - Centro de Ciências Sociais e Aplicadas, Universidade Regional de Blumenau, 1998.

DALFOVO, Oscar; AMORIM, Sammy Newton. **Quem tem informação é mais competitivo**. Blumenau: Acadêmica, 2000.

FREITAS, Henrique; LESCA, Humberto. **Competitividade empresarial na era da informação**. Revista de Administração, São Paulo, 1992.

FURLAN, José Davi; IVO, Ivonildo da Motta; AMARAL, Francisco Piedade. **Sistemas de Informações Gerencial - SIG**. São Paulo: Makron Books, 1994.

JAVA, Qual a diferença de Java e JavaScript. Disponível em: <http://java.com/pt_BR/download/faq/java_javascript.xml>. Acesso em: 19 ago. 2014.

LAUDON, Kenneth C.; LAUDON Jane P. **Gerenciamento de Sistemas de Informação**. 3. Ed. Rio de Janeiro : LTC, 2001.

LAUDON, Kenneth C; LAUDON, Jane Price. **Sistema da Informação com Internet**. [S.I.: s.n], 1999. 4.

MILANI, André. **MYSQL Guia do Programador**, Novatec Editora, Jan 8, 2007.

MYSQL Disponível em: <http://www.mysql.com/why-mysql/windows/>. Acesso em: 14 ago. 2014.

MYSQL WORKBENCH. **Mysql Workbench**, Disponível em: < http://dev.mysql.com/doc/relnotes/workbench/en/wb-news-5-2-35.html>. Acesso em 12 ago. 2014.

NETBEANS. **Netbeans**. Disponível em: <https://netbeans.org/community/articles/javaone/2014/netbeans-day-2014.html. Acesso em: 12 ago. 2014.

OLIVEIRA, Djalma. **Sistemas de informações gerenciais**: estratégias, táticas, operacionais. São Paulo: Atlas, 1998.

PHP. Disponível em: <http://php.net/releases/ >. Acesso em 08 ago. 2014.

PRESSMAN, Roger. **Engenharia de *software***, 5ª ed. Rio de Janeiro: McGraw-Hill, 2005.

REENSKAUG, Trygve. **The model-view-controller (MVC) its past and present**. (Artigo acadêmico - University of Oslo). 1979. 16 p. Acesso em: 03 Out. 2014.

REZENDE, Denis Alcides. **Planejamento de sistemas de informação e informática**. São Paulo: Atlas, 2003.

RODRIGUES, Leonel Cezar. Impactos dos sistemas de informação. **Caderno de Economia**, p. 2, 30 jun. 1996.

SAMPAIO, Cleuton. Guia do java enterprise edition 5: **desenvolvendo aplicações corporativas**. São Paulo: Brasport, 2007.

STAIR, Ralph M. **Princípios de sistemas de informação uma abordagem gerencial**. Rio de Janeiro: Livros Técnicos e Científicos, 1999.

STAIR, Ralph M. **Princípios de sistemas de informação**: uma abordagem gerencial. Rio de Janeiro: LTC, 1998.

VISUAL PARADIGM. **Visual Paradigm**. Disponível em: <http://www.visual-paradigm.com/>. Acesso em: 14 ago. 2014.