

UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
CÂMPUS CORNÉLIO PROCÓPIO
DIRETORIA DE GRADUAÇÃO E EDUCAÇÃO PROFISSIONAL
ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

HUGO HENRIQUE DERNEY DE AGUIAR

**SAMP: UM SISTEMA DE APOIO A FÁBRICAS DO SEGMENTO DE
MÓVEIS PLANEJADOS**

TRABALHO DE CONCLUSÃO DE CURSO

CORNÉLIO PROCÓPIO - PARANÁ

2015

HUGO HENRIQUE DERNEY DE AGUIAR

**SAMP: UM SISTEMA DE APOIO A FÁBRICAS DO SEGMENTO DE
MÓVEIS PLANEJADOS**

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Conclusão de Curso 2, do curso de Análise e Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná – UTFPR, como requisito para a obtenção do título de Tecnólogo.

Orientador: Prof. Dr. André Takeshi Endo

CORNÉLIO PROCÓPIO - PARANÁ

2015

FOLHA DE APROVAÇÃO

AGRADECIMENTOS

Agradeço ao meu orientador Prof. Dr. André Takeshi Endo, pela sabedoria com que me guiou nesta trajetória.

Aos meus colegas de sala.

A Secretaria do Curso, pela cooperação.

Gostaria de deixar registrado também, o meu reconhecimento à minha família, pois acredito que sem o apoio deles seria muito difícil vencer esse desafio.

Enfim, a todos os que por algum motivo contribuíram para a realização desta pesquisa.

RESUMO

AGUIAR, Hugo Henrique Derney de. **SAMP: Um sistema de apoio a fábricas do segmento de móveis planejados**. 2015. 58 f. Trabalho de Conclusão de Curso (Graduação) – Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná. Cornélio Procópio, 2015.

O uso de aparelhos móveis se torna cada vez comum para realizar tarefas do dia a dia. Devido a isso, existe uma técnica chamada de *Web Design Responsivo*, para que páginas Web possam se adaptar a resolução da tela. Que é um dos requisitos que uma aplicação deve possuir para se destacar no mercado. Por meio disso, foi pesquisado e não foram encontradas aplicações específicas para gerenciamento de fábricas de móveis planejados e que suprissem suas necessidades. Neste trabalho, foi desenvolvida uma aplicação que automatiza o fluxo de trabalho de empresas do segmento de móveis planejados, concretizando conceitos de responsividade.

Palavras-chave: Aplicações Web. *Web Design Responsivo*. Móveis planejados.

ABSTRACT

AGUIAR, Hugo Henrique Derney de. **SAMP: A support system for customized furniture segment factories**. 2015 58 f. Work Completion of course (Graduation) - Analysis and Systems Development. Federal Technological University of Paraná. Cornélio Procópio, 2015.

The adoption of mobile devices in daily tasks has grown habitual nowadays. As a consequence, a technique named Responsive Web Design takes place, which allows Web pages to fit in different screen sizes and resolutions. Full applications should hold a requirement set in order to make the difference at the current marketplace. The results of our research have indicated the absence of specific applications towards customized furniture factories' management. This paper presents an application that automates the work-flow of companies on the customized furniture segment.

Keywords: Web Application. Responsive Web Design. Customized Furniture.

LISTA DE FIGURAS

Figura 1: Exemplo de uma aplicação em diferentes dispositivos.....	16
Figura 2: Comando para a execução do SASS.....	19
Figura 3: Código CSS.....	19
Figura 4: Código utilizando SASS.....	20
Figura 5: Diagrama de Atividades – Fluxo de Trabalho.....	25
Figura 6: Diagrama geral de Caso de Uso.....	26
Figura 7: Diagrama de Classes.....	29
Figura 8: Diagrama de pacotes com a arquitetura geral da aplicação.....	30
Figura 9: Diagrama de pacotes com a arquitetura dos módulos.....	31
Figura 10: Classes Abstratas da aplicação.....	33
Figura 11: Caso de Uso Relacionar Material.....	34
Figura 12: Entidades do Caso de Uso Relacionar Material.....	35
Figura 13: Comando para gerar Entidades.....	35
Figura 14: MER para o Caso de Uso Relacionar Material.....	36
Figura 15: Diagrama de Classe - Controller e Model.....	37
Figura 16: Tela de login - Desktop.....	41
Figura 17: Tela de login – Dispositivo Móvel.....	41
Figura 18: Tela de Cadastro de Cliente – Desktop.....	42
Figura 19: Tela de Cadastro de Cliente – Dispositivo Móvel.....	43
Figura 20: Tela de listagem de pedidos – Desktop.....	43
Figura 21: Tela de cadastro de pedido – Desktop.....	44
Figura 22: Avaliar Design – Desktop.....	45
Figura 23: Agenda do Projetista – Desktop.....	45
Figura 24: Agenda do Projetista – Dispositivo Móvel.....	46
Figura 25: Listagem dos projetos – Desktop.....	46
Figura 26: Adicionar arquivos do design – Desktop.....	47
Figura 27: Projeto enviado para o cliente.....	47
Figura 28: Detalhes do projeto – Desktop.....	48
Figura 29: Detalhes do projeto – Dispositivo Móvel.....	48
Figura 30: Relacionar Material.....	49
Figura 31: Todos os projetos - Desktop.....	49
Figura 32: Todos os projetos – Dispositivo Móvel.....	50

Figura 33: Visualizar detalhes do projeto.....	50
Figura 34: Visualizar detalhes do projeto.....	51
Figura 35: Calcular preço final.....	52
Figura 36: Cadastrar pedido – Gerente.....	53
Figura 37: Cadastro de cliente.....	53

LISTA DE TABELAS

Tabela 1: <i>Sprints</i> realizadas para o desenvolvimento do SAMP.....	24
Tabela 2: Métricas do Projeto.....	40

LISTA DE SIGLAS

IBGE	Instituto Brasileiro de Geografia e Estatística
MDF	<i>Medium-Density Fireboard</i>
CSS	<i>Cascading Style Sheets</i>
MVC	<i>Model View Control</i>
UML	Linguagem Unificada de Modelagem
SASS	<i>Syntactically Awesome Style Sheets</i>
ORM	<i>Object Relational Mapper</i>
WDR	<i>Web Design Responsivo</i>

SUMÁRIO

1.INTRODUÇÃO.....	13
1.1 JUSTIFICATIVA.....	13
1.2 OBJETIVOS.....	14
2. FUNDAMENTAÇÃO TEÓRICA.....	16
2.1 APLICAÇÃO WEB.....	16
2.2 WEB DESIGN RESPONSIVO.....	16
2.3 SCRUM.....	18
2.4 RECURSOS UTILIZADOS.....	19
3. DESENVOLVIMENTO DO PROJETO.....	24
3.1 MÉTODO UTILIZADO.....	24
3.2 DIAGRAMAS.....	25
3.2.1 Diagrama de Atividades.....	25
3.2.2 Diagrama de Caso de Uso.....	27
3.2.3 Diagrama de Classes.....	30
3.3 ARQUITETURA DO SOFTWARE.....	31
3.4 DETALHES DE IMPLEMENTAÇÃO.....	33
3.5 VALIDAÇÃO DO SOFTWARE.....	39
3.6 MÉTRICAS.....	40
4. SISTEMA SAMP.....	41
4.1 GERAL.....	41
4.2 VISÃO DO CLIENTE.....	43
4.2.1 Solicitar Orçamento.....	43
4.2.2 Avaliar Design.....	46
4.3 VISÃO DO PROJETISTA.....	46
4.3.1 Realizar Design.....	46
4.3.2 Enviar para o cliente.....	48
4.4 VISÃO DO MARCENEIRO.....	48
4.4.1 Relacionar Material.....	48
4.5 VISÃO DO GERENTE.....	49
4.5.1 Realizar Agendamento.....	49
4.5.2 Escolher Marceneiro Responsável.....	50
4.5.3 Calcular Preço Final.....	51

4.5.4 Cadastrar Pedido de Orçamento.....	52
5.CONSIDERAÇÕES FINAIS.....	53
5.1 LIMITAÇÕES E DIFICULDADES ENCONTRADAS.....	53
5.2 TRABALHOS FUTUROS.....	54
REFERÊNCIAS.....	55

1. INTRODUÇÃO

1.1 JUSTIFICATIVA

Segundo dados do Instituto Brasileiro de Geografia e Estatística (IBGE, 2013), o percentual de pessoas no Brasil que acessaram a *Internet* cresceu de 20,9% em 2005, para 46,5% em 2011. Levando em consideração o crescimento da rede mundial de computadores, empresas usufruem dos benefícios que a Web oferece para desenvolver aplicações com o intuito de facilitar o gerenciamento de sua organização. A agilidade do acesso à aplicação é um dos atrativos que a Web oferece.

Segundo uma pesquisa realizada pela empresa F/NASCA SAATCHI & SAATCHI (2013), o acesso via celular cresceu 78,5% em apenas 20 meses, fato este que mostra um progresso surpreendente. Com isso surgiu à criação de aplicações Web, capazes de se adaptarem conforme a resolução do dispositivo em que está sendo utilizado, popularmente conhecido como *Web Design Responsivo*. Neste tipo de *design* são desenvolvidas páginas Web que alteram suas formas conforme as configurações do dispositivo em que está sendo visualizada (Souza, Igarashi, 2013, pag. 3).

Atualmente, as características que o *Web Design Responsivo* oferece são desejáveis em diferentes tipos de negócios. Dentre esses, existe o mercado de móveis planejados, caracterizado por oferecer a seus clientes a opção de construir móveis sob medida para um determinado imóvel. Por exemplo, móveis para apartamentos e lojas, tais como armários embutidos, guarda-roupas e camas entre outros. Assim cada projeto tem suas características conforme as medidas do ambiente e segundo as necessidades do cliente. Sendo assim, essas fábricas precisam de ferramentas computacionais que auxiliem as tarefas do dia a dia.

Foram encontrados diversos sistemas presentes no mercado, entre eles estão as seguintes aplicações: KAD Marcenaria, BMS Marcenaria e Starling Software. Existem algumas restrições nos sistemas presentes no mercado, podendo destacar, por exemplo, que a maioria destas aplicações ainda é Desktop. Outra restrição encontrada é que a maioria não fornece as funcionalidades necessárias para o

controle do projeto de uma marcenaria, alguns são sistemas desenvolvidos para se adaptarem a qualquer empresa, em muitos casos, não atendem as necessidades específicas de marcenarias. Por estes dados, foi definido o objetivo deste trabalho, que será tratado no próximo capítulo.

1.2 OBJETIVOS

O objetivo principal deste trabalho é a criação de uma aplicação Web que traz benefícios para os gerenciadores de empresas do segmento de móveis planejados, pois fornece a possibilidade de proporcionar uma forma de controlar o fluxo de trabalho da empresa, além de oferecer maior interação com o cliente. Foi tomada como base, uma empresa localizada no município de Bandeirantes – PR. Os usuários podem ter acesso ao sistema a partir de qualquer dispositivo móvel, além da flexibilidade das aplicações Web no qual os dados poderão ser armazenados na nuvem, sem correr o risco de que aconteça algo com o computador local os dados venham a se perder.

1.3 ORGANIZAÇÃO DO TEXTO

Este documento está organizado da seguinte forma: Capítulo 1 apresenta a introdução, que contém os subcapítulos com a justificativa do projeto, objetivos gerais e os problemas encontrados em aplicações presentes no mercado. O capítulo 2 apresenta a fundamentação teórica do trabalho. O capítulo 3 é dividido em subcapítulos e apresenta tópicos sobre o desenvolvimento da aplicação. O capítulo 4, mostra um manual de uso para os usuários. Enfim, o capítulo 5 apresenta as limitações e dificuldades e trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados os principais conceitos associados a este trabalho de diplomação. Levando como base as boas práticas que devem ser utilizadas para desenvolver uma aplicação de qualidade.

2.1 APLICAÇÃO WEB

Segundo Conallen (1999, apud GONÇALVES *et al.*, 2005, pag. 377), é considerada uma aplicação Web um *Web site* no qual é aplicada uma lógica de negócio e seu uso traz benefícios para quem utiliza a aplicação. Atualmente existem diversos tipos de *Web sites* presentes no mercado, entre eles estão os sites desenvolvidos com a finalidade de divulgação de uma empresa, que são os chamados *sites* institucionais. As lojas virtuais que fornecem vendas de produtos ou serviços, e as aplicações com o intuito de disponibilizar funcionalidades para auxiliar nas tarefas de empresas específicas. Muitos trabalhos enfocam somente o aspecto de desenvolvimento de software e outros no aspecto de design, com base em estética e mídia. Há aqueles que tomam como ponto, a produção de conteúdo informativo, arquitetura da informação e redação de texto para Web, mas as maiorias não levam em conta todo esse conjunto de finalidades (GONÇALVES *et al.*, 2005). O ideal é que a aplicação não seja somente funcional, ou somente bonita, por exemplo, o conjunto de características que irão torná-la diferenciada da maioria das aplicações presentes no mercado.

Uma aplicação pode possuir todas as qualidades possíveis, mesmo assim é preciso tomar cuidado na implantação de um sistema em uma empresa. Segundo SALVI *et al.* (2005, pg. 119), a introdução de um Sistema de Informação possui grande impacto comportamental e organizacional. Ela altera a maneira que os indivíduos e grupos se interagem, porque geralmente a informação é definida, acessada e usada para gerenciar recursos, pessoas e matéria prima da organização. Por fim, uma aplicação Web desde que bem projetada, pode trazer grandes benefícios para as empresas.

Pode-se incluir no conjunto de características que farão que a aplicação conquiste espaço no mercado, o conceito de *Web Design* Responsivo, que é a capacidade da aplicação para se adaptar a tela do dispositivo, e que será o assunto do próximo capítulo.

2.2 WEB DESIGN RESPONSIVO

Atualmente, com a chegada dos aparelhos eletrônicos portáteis, como por exemplo, *smartphone* e *tablet*, que têm como objetivo trazer mais praticidade aos usuários, a adaptação das aplicações para funcionarem em qualquer dispositivo mantendo sua funcionalidade e acessibilidade, vem sendo cada vez mais requisitado no mercado.

Segundo Souza e Igarashi (2013, pag. 3), a navegação em aplicações Web por dispositivos móveis se vê prejudicada devido à dimensão do visor dos dispositivos, que são inferiores aos monitores nos quais grande parte das aplicações são apresentadas. Por este motivo o *Web Design Responsivo* tem o objetivo de moldar a aplicação, alterando o tamanho das imagens e dos elementos, posicionando-os de uma maneira funcional conforme o dispositivo. No exemplo apresentado na Figura 1 é exposto como o mesmo site fica em um *smartphone* com resolução de 320 pixels(px), em um tablet na vertical com resolução de 768px e na horizontal com a resolução de 1024px, e em um notebook representando um computador desktop com a resolução de 1068px utilizando o conceito de responsividade.

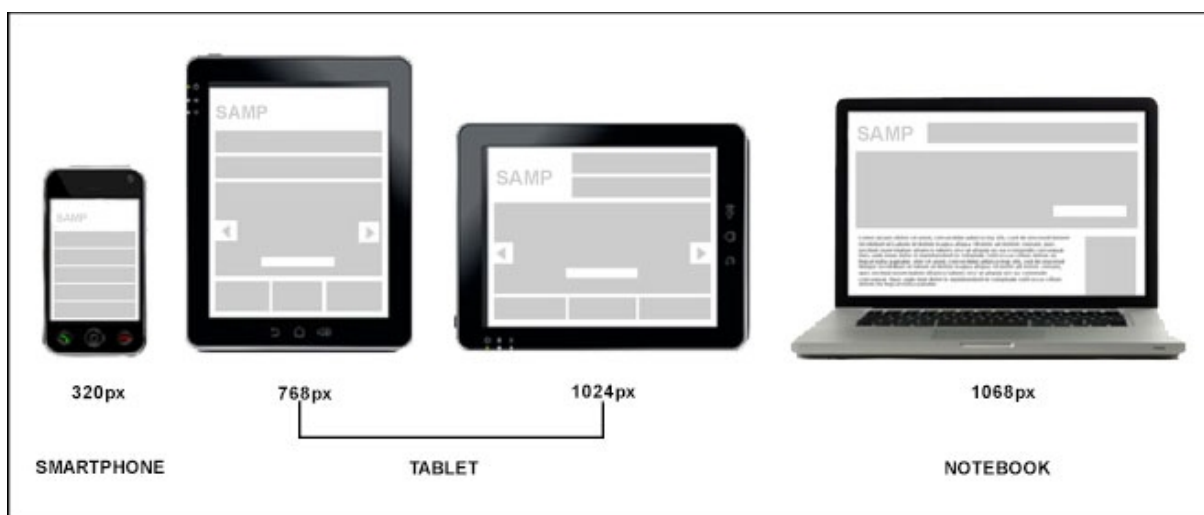


Figura 1: Exemplo de uma aplicação em diferentes dispositivos.

Fonte: Adaptado de SOUZA e IGARASHI (2013, pag. 3).

O *Web Design Responsivo* é classificado em três principais etapas:

1. *Layouts* fluídos, no qual os elementos não terão largura fixa para que possam se adaptar conforme a resolução do dispositivo.

2. A segunda etapa é caracterizada por possuírem imagens e recursos flexíveis, sendo assim, eles se adaptarão conforme seu elemento pai.
3. A terceira etapa é a utilização de *Media Queries* que permite conforme a resolução, mostrar, ocultar e alterar elementos para que fiquem de acordo com o monitor do usuário.

Por meio destas etapas é desenvolvida uma aplicação responsiva que oferece uma maior acessibilidade e usabilidade para o usuário. Também é necessário uma metodologia para utilizar no desenvolvimento da aplicação, esta metodologia que será assunto do próximo capítulo.

2.3 SCRUM

Segundo Schawaber e Sutherland (2013, pag. 4), Scrum é um *framework* baseado na teoria do empirismo, em que as tomadas de decisões são definidas por meio de experiências. Essa teoria segue três pilares que apoiam a implementação de controle de processo:

- **Transparência:** Os observadores devem compartilhar o mesmo entendimento do que vai ser visto. Por exemplo, eles devem ter o mesmo conceito de “pronto”.
- **Inspeção:** Os usuários devem frequentemente realizar verificações sobre o progresso dos artefatos, com intenção de detectar variações. Porém, essa inspeção não deve ser realizada de modo contínuo, a ponto de atrapalhar a execução das tarefas.
- **Adaptação:** Se o inspetor verificar alterações relevantes no processo, devem ser estudadas novas alternativas para realizar ajustes o mais breve possível.

Pode-se destacar as seguintes principais características do Scrum, segundo Schawaber e Sutherland:

- O Time Scrum, é composto pelo *Product Owner* que é o dono do produto, responsável por valorizá-lo, a Equipe de Desenvolvimento que é responsável por implementar o produto e tem a responsabilidade de entregar um

incremento no final de cada *Sprint* e o *Scrum Master* que é o responsável por verificar se as técnicas de Scrum estão realmente sendo aplicadas, ajudando também os que estão fora do Time Scrum para interagirem com os integrantes. O time é auto reorganizável, no qual escolhem qual a melhor forma de reorganizarem seu trabalho e seus profissionais são multifuncionais, em que possuem todas as qualidades necessárias para completar o serviço sem depender de terceiros.

- O *Product Backlog* que se trata de uma lista ordenada de artefatos que serão necessários no produto. O responsável por gerar essa lista é o *Product Owner*.
- *Sprint*, que pode ser considerada o item em qual o Scrum se baseia, é um tempo definido, normalmente de cerca de um mês, que é trabalhado para entregar um incremento do produto. Após o fim do ciclo, outro é iniciado momentaneamente, até o produto final estar pronto. As *Sprints* são compostas por reuniões de planejamento, reuniões diárias, trabalho de desenvolvimento, e uma revisão.
- *Sprint Backlog* são os itens selecionados a partir do *Backlog* do Produto para fazerem parte da *Sprint* a fim de conseguir realizar a entrega do próximo incremento.

2.4 RECURSOS UTILIZADOS

Neste Subcapítulo, serão descritas as tecnologias e ferramentas utilizadas, para o desenvolvimento desta aplicação Web.

- *Hyper Text Markup Language* (HTML) (W3C, 2015), é uma linguagem de marcação utilizada para desenvolver páginas Web. Basicamente o HTML é responsável por definir a estrutura das páginas que serão desenvolvidas.
- *Cascading Style Sheets* (CSS) (W3C, 2015), são folhas de estilo em cascata que são responsáveis por adicionar estilos (cores, fontes, espaçamentos) para documentos Web, deixando-as visualmente elegante.

- *Javascript* (W3C, 2015), é uma linguagem de *script* utilizada para manipulação de elementos HTML e CSS para trazer uma maior interação com o usuário.
- *Syntactically Awesome Style Sheets* (SASS) (CATIN et. al, 2015) é uma extensão do CSS que apresenta recursos para desenvolver códigos aninhados e de forma mais organizada. SASS foi desenvolvido em Ruby, uma linguagem dinâmica e *open source* com foco na simplicidade e na produtividade. Na Figura 2 é ilustrado como utilizá-lo após instalado, neste caso, em um sistema operacional Linux. É utilizado o comando “`sass - -watch`” e logo em seguida o nome da pasta no qual constam os arquivos com extensão `scss`, neste caso a pasta `scss`, depois a pasta em que será salvo dos arquivos com extensão `css`, representada pela pasta `css`, como parâmetro opcional foi utilizado o *Style* que define que os arquivos `css` serão gerados de forma minificada, prezando um carregamento mais veloz para a aplicação.

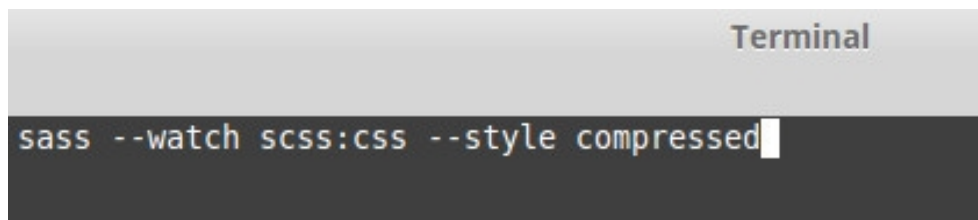
A screenshot of a terminal window with a grey title bar labeled "Terminal". The terminal content shows the command `sass --watch scss:css --style compressed` followed by a white cursor block at the end of the line.

Figura 2: Comando para a execução do SASS
Fonte: Autoria própria

As Figuras 3 e 4 mostram a diferença entre dois códigos que atingem o mesmo objetivo, a Figura 3 utilizando CSS e a Figura 4 desenvolvido utilizando SASS. Importante mencionar que o arquivo SCSS que é correspondente ao SASS, não é o que será chamado nas páginas Web, e sim o arquivo CSS que será gerado, que já vem minificado, prezando um carregamento mais veloz para a aplicação.

```
body header div.exemplo {  
  float: left;  
  display: inline-block;  
}  
  
body header div.exemplo p {  
  text-align: center;  
}
```

Figura 3: Código CSS
Fonte: Autoria própria

```
body {  
  header {  
    div.exemplo {  
      float: left;  
      display: inline-block;  
    }  
  
    p {  
      text-align: center;  
    }  
  }  
}
```

Figura 4: Código utilizando SASS
Fonte: Autoria própria

- *Hypertext Preprocessor* (PHP) (The PHP Group, 2015) é uma linguagem de *script* específica para desenvolver aplicações Web, rápidas e eficazes. Utilizada para desenvolver desde pequenas até grandes aplicações, é uma linguagem interpretada pelo servidor Web.
- Zend Framework (Zend Framework, 2015) é um poderoso *Framework* de PHP que possui recursos para auxiliar o desenvolvimento aplicações Web. Suas principais características são:
 - o Framework *Model-View-Controller* (MVC) que separa o desenvolvimento em *Model*, que fica responsável por realizar operações com o banco de dados, *View*, que são onde ficam os arquivos de interface que será apresentado para o usuário, e *Controller* que fornece a regra de negócio, essa separação permite que o sistema

- tenha um baixo acoplamento facilitando o desenvolvimento e manutenção do código fonte.
- o Possibilita o desenvolvimento em módulos, que auxilia com o reaproveitamento de código e manutenção do sistema.
 - o Segue o paradigma de orientação a objetos que torna o código mais organizado e otimizado. Utiliza também o conceito de namespaces que poupa a utilização de “*require*” nos arquivos que utilizarão código de outros arquivos.
 - o Fornece uma ótima estrutura para utilização de formulário, contendo suas devidas classes que possibilitam adicionar validações, e caso o formulário enviado não seja válido, a página continua no formulário com os dados preenchidos, porém com suas devidas informações sobre erros.
- MySQL (Oracle Corporation et. al, 2015) é um banco de dados de código aberto que possui grande desenvoltura em sistemas Web. Será utilizada a ferramenta de Mysql Workbench gerenciar o banco de dados.
 - *PhpStorm* (JetBrains, 2015) é uma IDE de PHP sofisticado fornecendo assistência de codificação de alto nível e suporte para as principais ferramentas e frameworks disponíveis no mercado. Suas principais características são:
 - o Autocomplete.
 - o Análise de qualidade de código, verificando a existência de códigos repetidos, inutilizados ou com más práticas de desenvolvimento é adicionado um aviso.
 - o Acesso à base de dados, para visualizar e manipular informações das tabelas.
 - o Atalhos do teclado para maior agilidade no desenvolvimento.
 - o Depurador de código.
 - o Integração com o versionador de código GIT.

- o Acesso ao terminal do sistema operacional.
 - o Suporte ao composer, que é um gerenciador de dependências do PHP.
- Bootstrap (Bootstrap, 2015) é um *Framework* de HTML, CSS e *Javascript* desenvolvido para criar aplicações Web que se adaptem a qualquer tipo de dispositivo móvel. Também um ótimo recurso para auxiliar no *design* das aplicações.
- GIT (Git, 2015) é um controlador de versão de código aberto, capaz de suportar de pequenos até grandes projetos. Existe suporte para utilizá-lo na interface, porém neste trabalho foi utilizado terminal Linux para o gerenciamento do projeto. Para utilizá-lo basta instalar, abrir o terminal e acessar a pasta do projeto, depois do repositório iniciado é basicamente utilizado os seguintes três comandos para controlar as versões: “git add x”, adiciona novos arquivos para serem adicionados ao projeto. “git commit -m 'texto de informação’”, este comando concretiza as alterações realizadas e prepara o ambiente para ser atualizado no repositório. E “git push origin nomeorigin” que realiza a atualização do repositório. Existem vários outros comandos que o Git fornece, porém esse foram os mais utilizados para o gerenciamento deste trabalho.
- Astah (Astah, 2015) é uma ferramenta utilizada para agilizar o processo de criações de diagramas baseados no UML. Fornece várias opções para criação de gráficos, na versão gratuita é possível criar os seguintes diagramas: Classes, Caso de Uso, Máquina de estado, Atividade, Sequência, Comunicação, Componente, Desenvolvimento e Estrutura de composição.
- Doctrine (Doctrine Team, 2015) é uma junção de várias bibliotecas de PHP focadas principalmente no armazenamento de bancos de dados e mapeamentos de objetos. Auxiliando com relacionamentos Objeto-Relacional. O projeto foi inspirado em conceitos do Hibernate ORM que é utilizado pela linguagem de programação Java. Doctrine utiliza anotações para que possa visualizar quais classes que devem ser geradas como tabelas. Possui também suporte para realizar operações com o banco de dados.

- Composer (Composer, 2015) é um gerenciador de PHP, basta definir as bibliotecas que serão utilizadas que ele faz todo o trabalho de instalar e atualizar os pacotes. É também usado utilizando o terminal, no Linux.

3. DESENVOLVIMENTO DO PROJETO

3.1 MÉTODO UTILIZADO

A metodologia utilizada para o gerenciamento do processo de criação desta aplicação Web foi o Scrum. Ele foi adaptado e algumas práticas foram adotadas para o aluno utilizar acompanhado de seu orientador.

O orientador deste trabalho fez o papel do *Scrum Master*, garantindo que o *framework* fosse aplicado corretamente e o aluno fez o papel da equipe de desenvolvimento, neste caso, adaptado para uma só pessoa.

O *Product Backlog* neste trabalho é representado pelos diagramas de atividades, diagramas de caso de uso e diagramas da UML em geral. No Scrum o *Product Owner* que é o responsável pela Equipe de Desenvolvimento, tem o papel de construir o *Product Backlog*, neste caso como não existe um *Product Owner*, o *Product Backlog* foi realizado pela Equipe de Desenvolvimento, que neste caso é representado pelo aluno. Inicialmente foi gerado um diagrama de atividades para entender o fluxo de trabalho da empresa, por meio deste fluxo foi possível analisar o que poderia ser adicionado nesta aplicação de forma que inicialmente não afetasse drasticamente a rotina da empresa, e que auxiliaria no gerenciamento de fluxo de trabalho. Após definido o diagrama de atividades, foi elaborado o diagrama de caso de uso, e que cada caso de uso foi utilizado como item do *Backlog* da *Sprint*. A duração média do tempo da *Sprint* foi definida como duas semanas, sendo realizada pela Equipe de Desenvolvimento e o *Scrum Master*.

Também ao final das *Sprints* eram realizadas a chamada *Sprint Review*, a fim de mostrar um demo do novo incremento do produto ao *Scrum Master* que verificava se haveriam correções necessárias. Também foi realizada uma *Sprint Review* com um marceneiro representando o *stakeholder*, mas, não foi possível realizar a reunião no final do ciclo de cada *Sprint*, devido à dificuldade encontrada ao entrar em contato com os *stakeholders*.

Na Tabela 1, é apresentado o cronograma das *Sprints* com informações sobre quais casos de uso foram escolhidos para trabalhar na *Sprint*. A coluna de correções mostra os reparos realizados na transição de uma *Sprint* para outra. Esses reparos

são encontrados pela *Sprint Review*, realizada ao final do ciclo de cada *Sprint* e por fim, a quantidade de semanas que foram utilizadas para entregar cada incremento.

Tabela 1: Sprints realizadas para o desenvolvimento do SAMP

SPRINT	CASOS DE USO	CORREÇÕES	SEMANAS
1	CRUD Usuário		6
2	Login\Logout	Correções em CRUD Usuário	6
	Solicitar Orçamento		
	Atualizar Dados		
	Efetuar Cadastro		
	Acompanhar Pedido		
3	CRUD Projeto		5
	Realizar Agendamento		
	Realizar Design		
4	Enviar Projeto para o Cliente		1
	Aprovar Projeto		
	Escolher marceneiro responsável		
5	Relacionar Material		2
	Estimar prazo de entrega		
	Calcular preço final		
6	Enviar proposta p/ cliente	Completar Cadastro do Projeto	2
	Analisar Proposta	Mostrar só eventos do projetista – Agendar Visita	
	Agendar Montagem	Permitir todos os arquivos – Design	

Fonte: Autoria própria

Pode-se observar que nem todas as *Sprints* seguiram o prazo de duas semanas que havia sido planejado. No desenvolvimento das primeiras *Sprints* foram encontradas dificuldades para se trabalhar com o Zend Framework, acarretando aumento do tempo de desenvolvimento. A partir da quarta *Sprint*, é possível observar uma estabilidade na quantidade de semanas.

3.2 DIAGRAMAS

3.2.1 Diagrama de Atividades

Na Figura 5 é apresentado o fluxo de trabalho utilizando o diagrama de atividades, em que passo a passo do fluxo de atividade que foi analisado e definido e que poderia ser automatizado da forma que não prejudicasse o processo de trabalho da empresa, desde a entrada, design, desenvolvimento, até a entrega do projeto. O fluxo de trabalho foi baseado na empresa que está servindo de base para esta monografia. Cada raia no diagrama representa um tipo do usuário do sistema:

cliente, gerente, projetista e marceneiro. Cada atividade representa uma ou mais tarefas que devem ser realizadas para prosseguir com o ciclo. A sequência do ciclo é definida pelas flechas, que também apontam para um diamante, que representa uma estrutura de decisão, ou seja, a próximo passo só é definido conforme o resultado da atividade anterior.

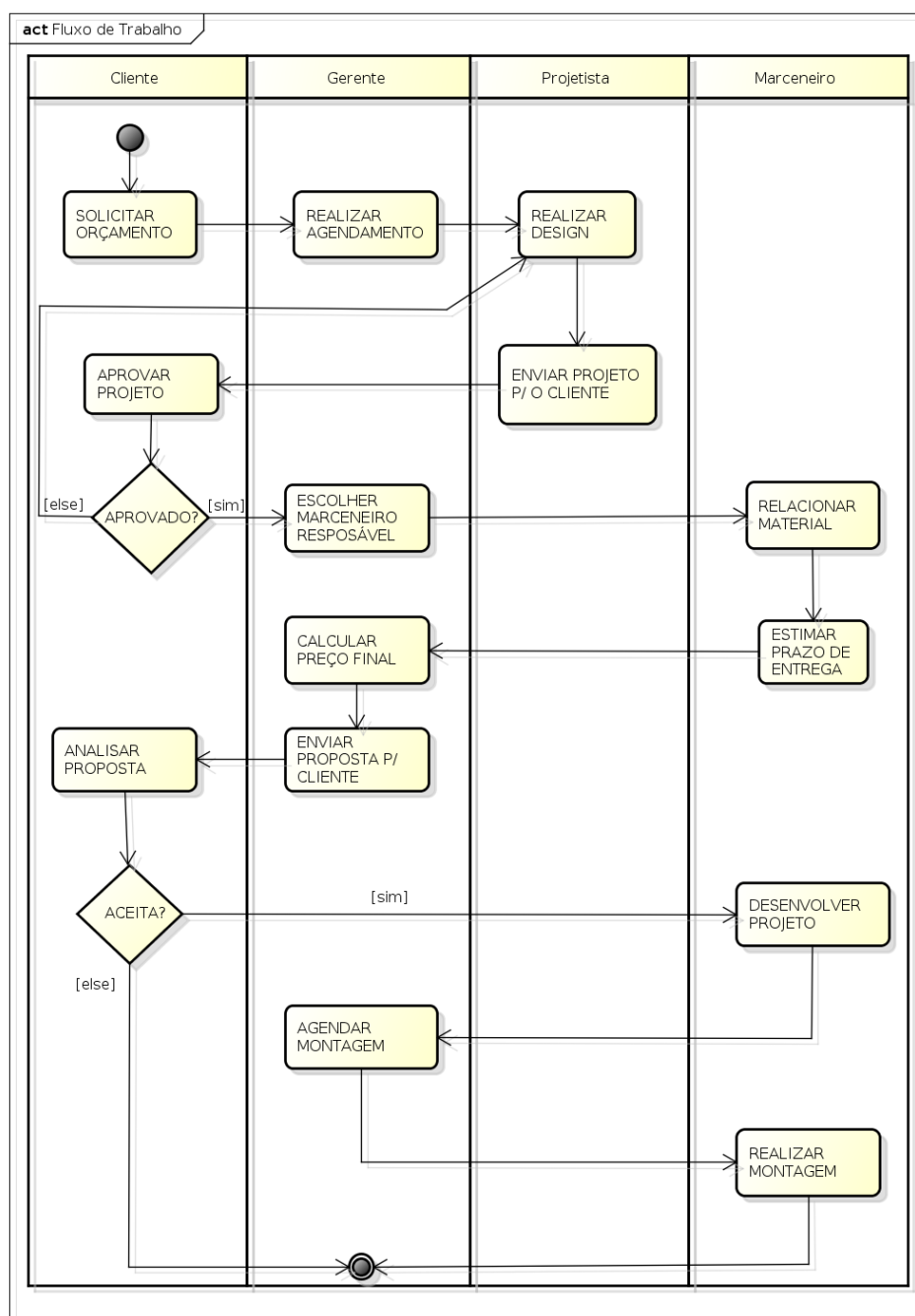
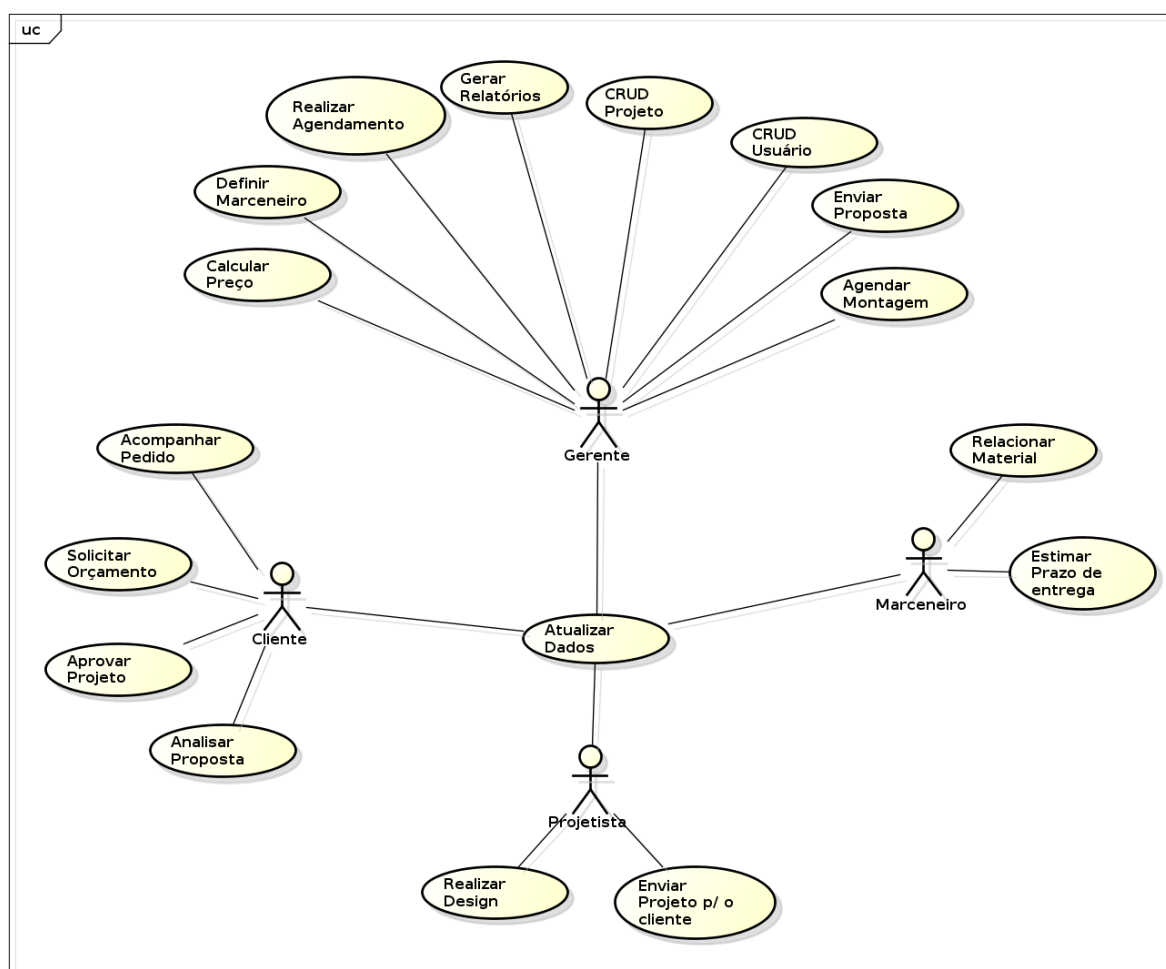


Figura 5: Diagrama de Atividades – Fluxo de Trabalho

Fonte: Autoria própria

3.2.2 Diagrama de Caso de Uso

Na Figura 6 é apresentado um diagrama geral de caso de uso, de modo que seja demonstrada uma visão geral da aplicação, contendo quatro tipos de usuários, que são capazes de executar suas devidas atividades de forma automatizada. Cada balão representa uma atividade que pode ser composta de uma ou mais tarefas. Foi optado por não realizar a descrição detalhada dos casos de usos, pelo fato que havia contato direto com os integrantes da empresa, que foi tomada como base do projeto.



powered by Astah

Figura 6: Diagrama geral de Caso de Uso.

Fonte: Autoria própria

Segue a lista dos casos de uso com uma breve descrição:

- **CRUD Usuário:** Criar, editar e excluir usuários, só disponível para o Gerente.
- **Solicitar orçamento:** O cliente solicita um pedido de orçamento pela Web ou diretamente com o Gerente. As informações do pedido são armazenadas e ficam aguardando análise.
- **Acompanhar pedido:** O cliente pode acompanhar o status do pedido, que são rotulados como: aguardando aprovação, aceito ou cancelado. O gerente tem o privilégio de adicionar status para o pedido.
- **CRUD projeto:** Criar, editar ou excluir projetos. Um projeto pode ser criado do zero, ou a partir de um pedido de orçamento.
- **Realizar agendamento:** Agendar eventos que podem ser relacionados com reuniões com clientes ou visitas, que o projetista deverá realizar no cliente para tirar medidas do ambiente.
- **Realizar design:** A aplicação fornece a possibilidade para o projetista adicionar os desenhos do projeto na aplicação, como formato de imagem que também será visível para o cliente, ou formato utilizado pela ferramenta de desenvolvimento do projetista.
- **Enviar projeto para o cliente:** Após o projetista ter realizado o *design*, ele deve enviar o projeto para o cliente analisar.
- **Aprovar projeto:** O cliente recebe o *design* e visualiza as imagens no próprio sistema e após analisar, aprova ou rejeita. Para mais esclarecimentos sobre o projeto, a fábrica ainda manterá o procedimento, realizando reuniões pessoalmente com o cliente. A aplicação será só uma forma de complementar e documentar o procedimento.
- **Definir marceneiro:** O gerente deve selecionar o marceneiro que trabalhará no projeto.
- **Relacionar material:** O marceneiro adiciona uma lista de itens que serão necessários para o desenvolvimento do projeto. Essa lista é editável, pode

adicionar e remover itens. Após os itens adicionados pelo marceneiro, o gerente poderá adicionar o valor unitário do item.

- **Estimar prazo de entrega:** O marceneiro estima um prazo de entrega para o projeto, assim o gerente pode desenvolver o orçamento baseado no tempo de desenvolvimento.
- **Calcular preço:** O gerente adiciona o valor de mão de obra, o sistema faz o cálculo da mão de obra mais o valor total da relação de material, gerando o valor final do produto.
- **Enviar proposta:** Após o valor final do produto definido, o gerente deve enviar a proposta para o cliente.
- **Analisar proposta:** O cliente analisa a proposta que foi liberada pelo gerente e com isso, define se aceitará ou rejeitará a proposta.
- **Agendar montagem:** O gerente agenda a montagem para o projeto, após a conclusão do desenvolvimento, é automaticamente marcado na agenda do marceneiro responsável pelo projeto.
- **Gerar relatórios:** Responsável por gerar relatórios de algumas seções da aplicação e conforme o privilégio do usuário. Por exemplo, após a relação de material for relacionada pelo marceneiro, é possível gerar um PDF para que através daqueles itens, o gerente conseguir realizar orçamentos de materiais com empresas do ramo.

3.2.3 Diagrama de Classes

Na Figura 7, é apresentado o Diagrama de Classes, representando as entidades que foram necessárias para a aplicação. Os atributos foram marcados como *protected* (#) para que o Doctrine pudesse manipulá-los.

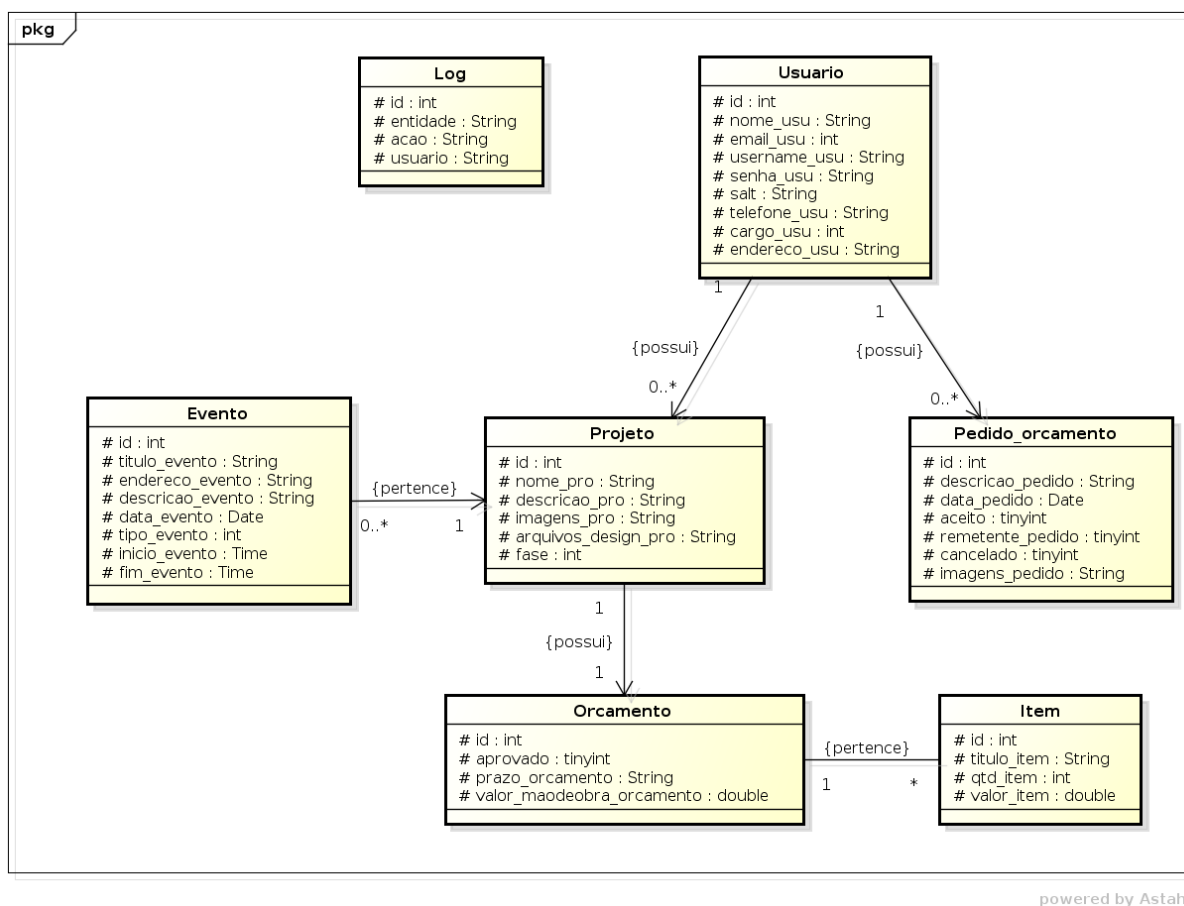
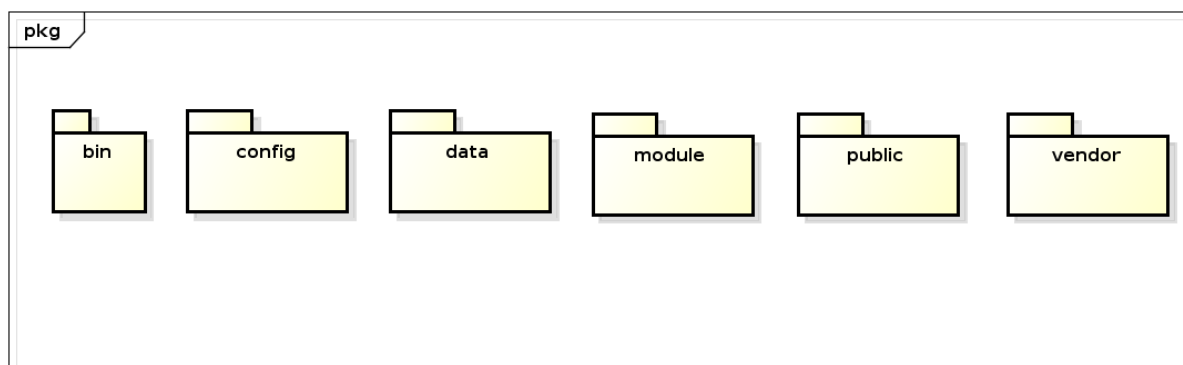


Figura 7: Diagrama de Classe

Fonte: Autoria própria

3.3 ARQUITETURA DO SOFTWARE

Na Figura 8, é apresentado um diagrama de Pacotes com a arquitetura geral do sistema e logo abaixo, uma breve descrição sobre cada pacote. Cada pacote representa uma pasta do sistema.



powered by Astah

Figura 8: Diagrama de pacotes com a arquitetura geral da aplicação

Fonte: Autoria própria

A seguir, uma breve descrição de cada pacote:

- **Bin:** Contém arquivos de configuração gerados pelo Doctrine.
- **Config:** Pasta da arquitetura do Zend Framework. Contém arquivos de configuração essenciais para o funcionamento do sistema, inclusive configurações de acesso ao banco de dados.
- **Data:** Pasta utilizada para armazenar informações, normalmente somente pelo sistema, já que esta pasta não é de acesso público.
- **Module:** Contém todos os módulos desenvolvidos na aplicação.
- **Public:** Contém todos os arquivos de acesso público, por exemplo, arquivos visuais como CSS, Javascript e imagens. É também a pasta a qual o servidor deve apontar, pois é nela que está configurada a página principal da aplicação.
- **Vendor:** Pasta utilizada pelo Composer para armazenar o módulo do Zend Framework e todos os módulos adicionais utilizados pela aplicação.

Uma parte muito importante da aplicação, são os módulos, e para ficar mais claro sobre sua estrutura, foi implementado o diagrama de pacotes, Figura 9, utilizando o módulo Base como exemplo.

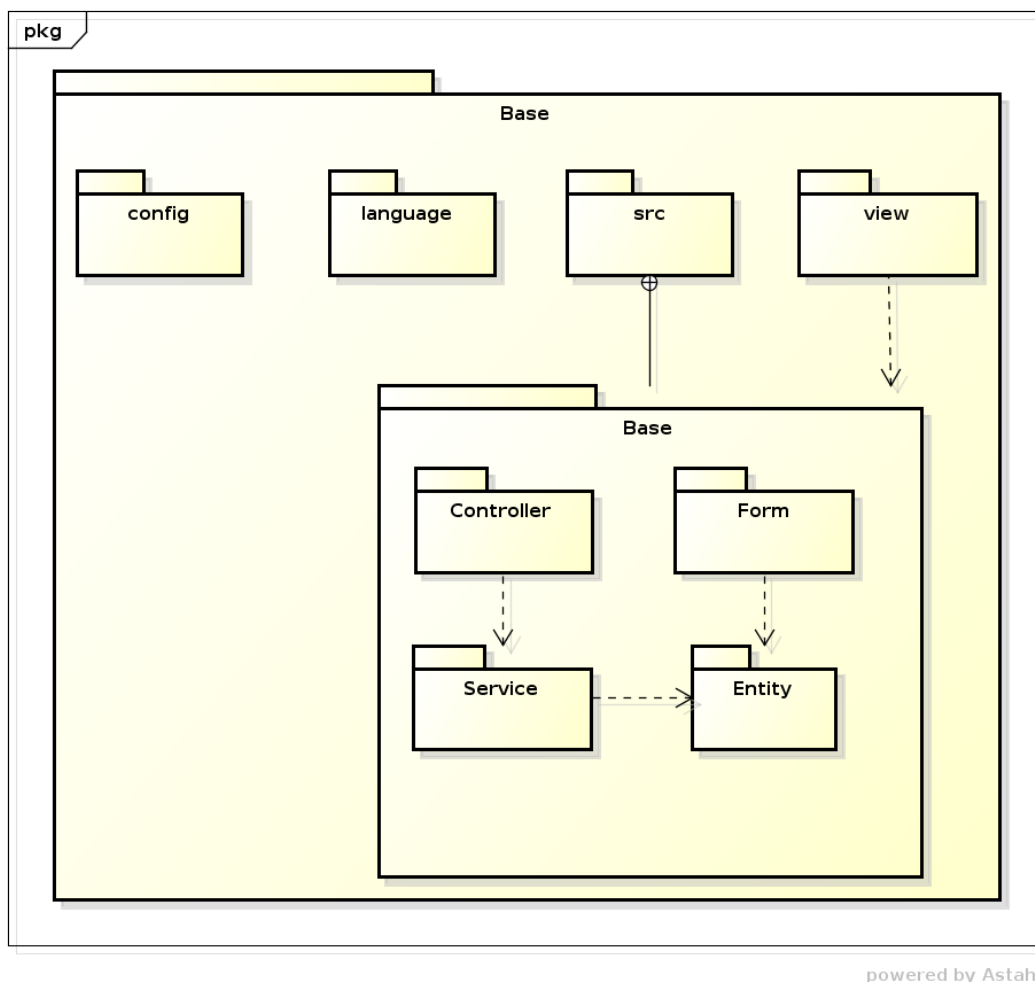


Figura 9: Diagrama de pacotes com a arquitetura dos módulos

Fonte: Autoria própria

A seguir, uma breve descrição sobre a estrutura dos módulos:

- **Config:** Contém o arquivo de configuração do módulo, essencial para seu funcionamento.
- **Language:** Contém os arquivos de idioma utilizados pelo Zend Framework. Esta aplicação não foi desenvolvida com suporte a múltiplos idiomas, por esta razão a pasta só possui o arquivo de suporte a língua portuguesa.

- **View:** Pasta de visão do módulo, no qual ficam todos os arquivos do formato PHTML que é utilizado pelo Zend Framework.
- **SRC:** Contém uma pasta com o mesmo nome do módulo, e dentro dela, pastas de implementações que não estão em todos os módulos, só no caso do módulo utilizar os recurso das pastas. Em geral, o diretório pode conter as seguintes pastas:
 - **Controller:** Armazena as classes que possuem a regra de negócio da aplicação.
 - **Entity:** Armazena as classes de entidades, que são desenvolvidas com anotações do Doctrine para gerar as tabelas no banco de dados MySQL.
 - **Service:** Representa a separação *Model* do conceito de MVC, possui classes responsáveis por realizar operações com a base de dados.
 - **Form:** Armazena as classes de formulários que são criados a partir da estrutura do Zend Framework.

Esta é a arquitetura que a aplicação está utilizando, com a finalidade de mostrar um pouco mais sobre o desenvolvimento, o próximo tópico traz mais detalhes sobre a implementação do projeto.

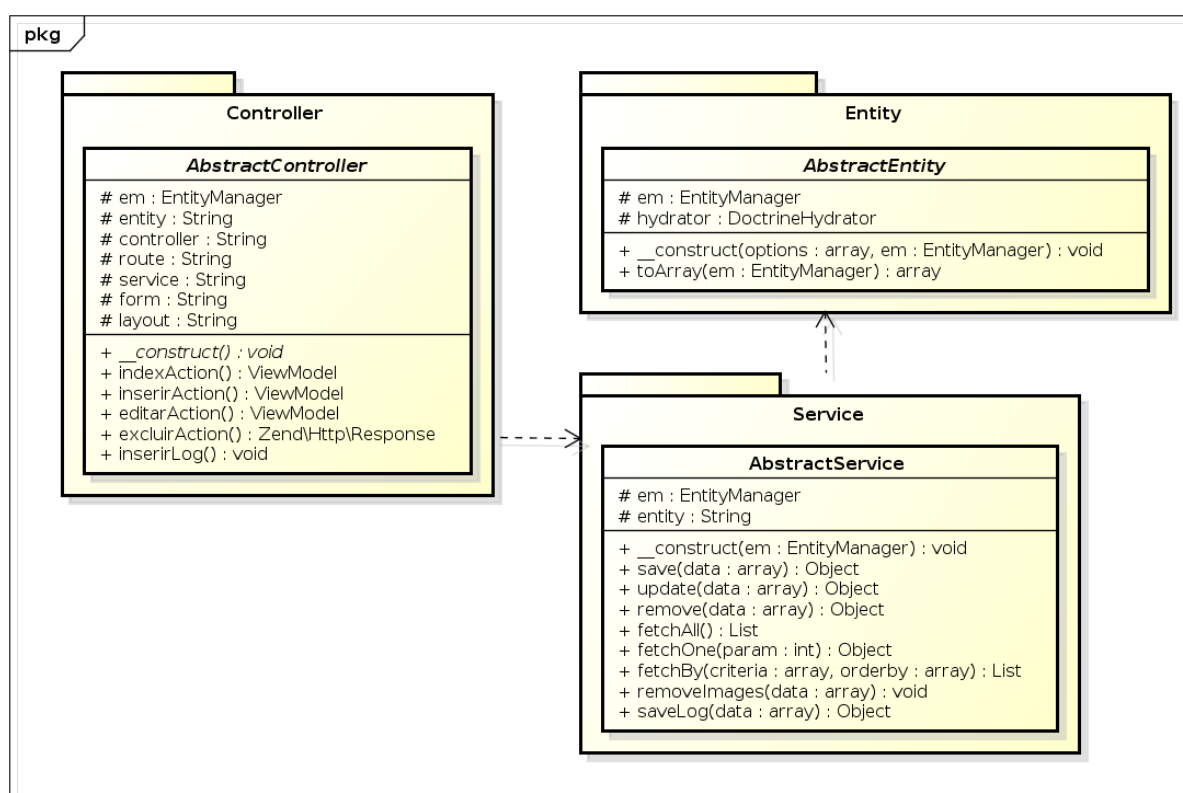
3.4 DETALHES DE IMPLEMENTAÇÃO

Para o desenvolvimento desta aplicação, foi criado um módulo nomeado como “Base” no Zend Framework. Este módulo tem o objetivo de fornecer classes abstratas, que disponibilizam os métodos mais utilizados no sistema. Neste módulo também se encontram as classes que são utilizadas por vários módulos da aplicação. Basicamente, foram desenvolvidas três classes Abstratas, que estão separadas em pacotes diferentes, respeitando o modelo MVC, são eles:

- **Controller:** Neste pacote foi implementada a classe `AbstractController`, que estende a classe `AbstractActionController`, de uso obrigatório nas classes controladoras do Zend Framework.

- **Entity:** Neste pacote foi implementada a classe `AbstractEntity`, responsável por implementar recursos que o Doctrine fornece para construir objetos. Ela transforma uma matriz de dados no objeto que foi instanciado e também transforma objeto em uma matriz de dados.
- **Service:** Neste pacote foi desenvolvida a classe `AbstractService`. Responsável por realizar operações com o banco de dados, o pacote `Service` representa a separação *Model* do modelo MVC.

Na Figura 10, foi elaborado um diagrama de classes mostrando a estrutura das classes Abstratas, dentro de seus determinados pacotes, citados anteriormente.



powered by Astah

Figura 10: Classes Abstratas da aplicação
Fonte: Autoria própria

Para melhor entendimento sobre o processo de desenvolvimento, será realizado demonstrações sobre o Caso de Uso Relacionar Material, conforme a Figura 11.

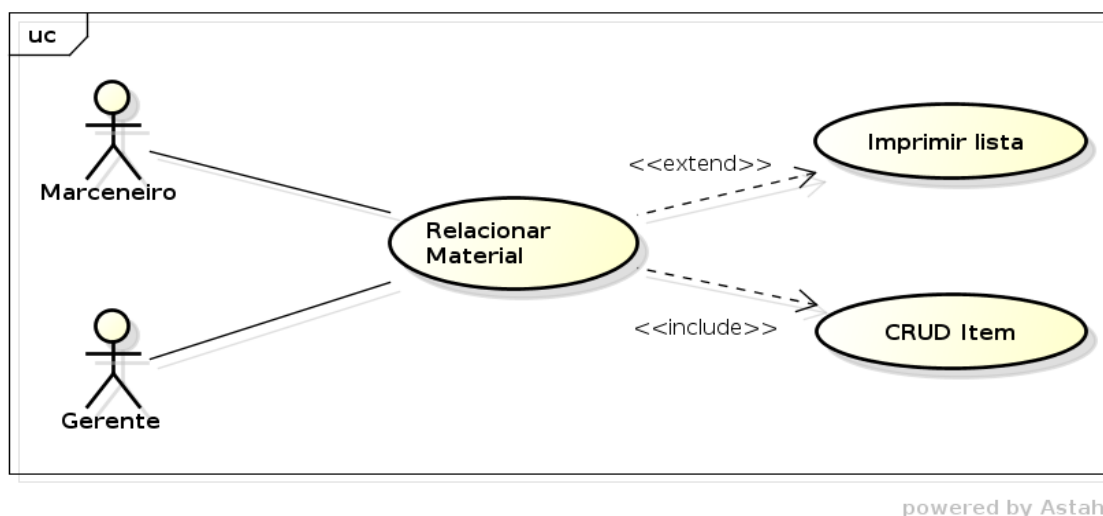


Figura 11: Caso de Uso Relacionar Material
Fonte: Autoria própria

Participam deste Caso de Uso dois atores, o Gerente e o Marceneiro, e que ambos tem seus privilégios. O Marceneiro pode adicionar e remover itens da lista de materiais, porém ele não tem a possibilidade de adicionar valor de unidade do item adicionado. O Gerente não só consegue adicionar valor, como também adicionar e remover itens.

Duas entidades são necessárias para a criação deste caso de uso, a entidade Orçamento e Item, a entidade Orçamento não vai ser manipulada no desenvolvimento deste Caso de Uso, porém, é importante mostrar que os itens são dependentes da entidade Orçamento. Na Figura 12, é possível observar o diagrama de classe destas entidades.

Nessas entidades são adicionadas anotações do Doctrine, para serem mapeadas para o banco de dados. Para isso, é necessário executar um comando no terminal. Este comando está representado na Figura 13, e deve ser executado dentro da pasta bin do projeto.

Utilizando um recurso do PHPStorm, é possível verificar se a tabela foi gerada corretamente, através de um diagrama gerado das tabelas selecionadas. O resultado do diagramas para as tabelas Orçamento e Item é mostrado na Figura 14.

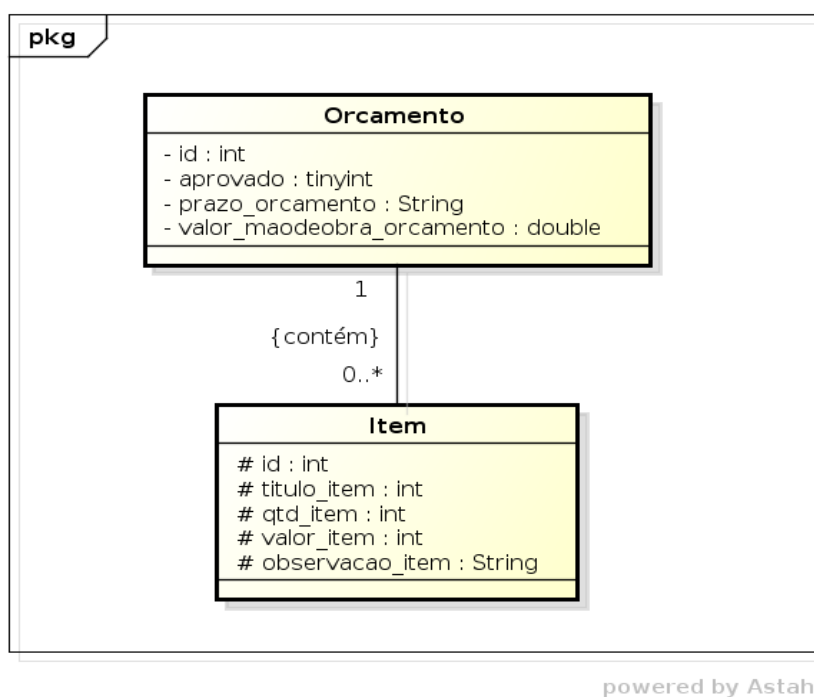


Figura 12: Entidades do Caso de Uso Relacionar Material
 Fonte: Autoria própria

```
php doctrine-module orm:schema-tool:update --force
```

Figura 13: Comando para gerar Entidades
 Fonte: Autoria própria

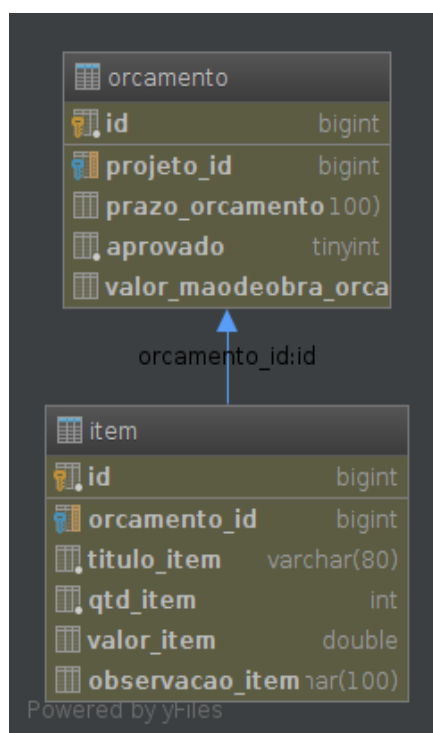
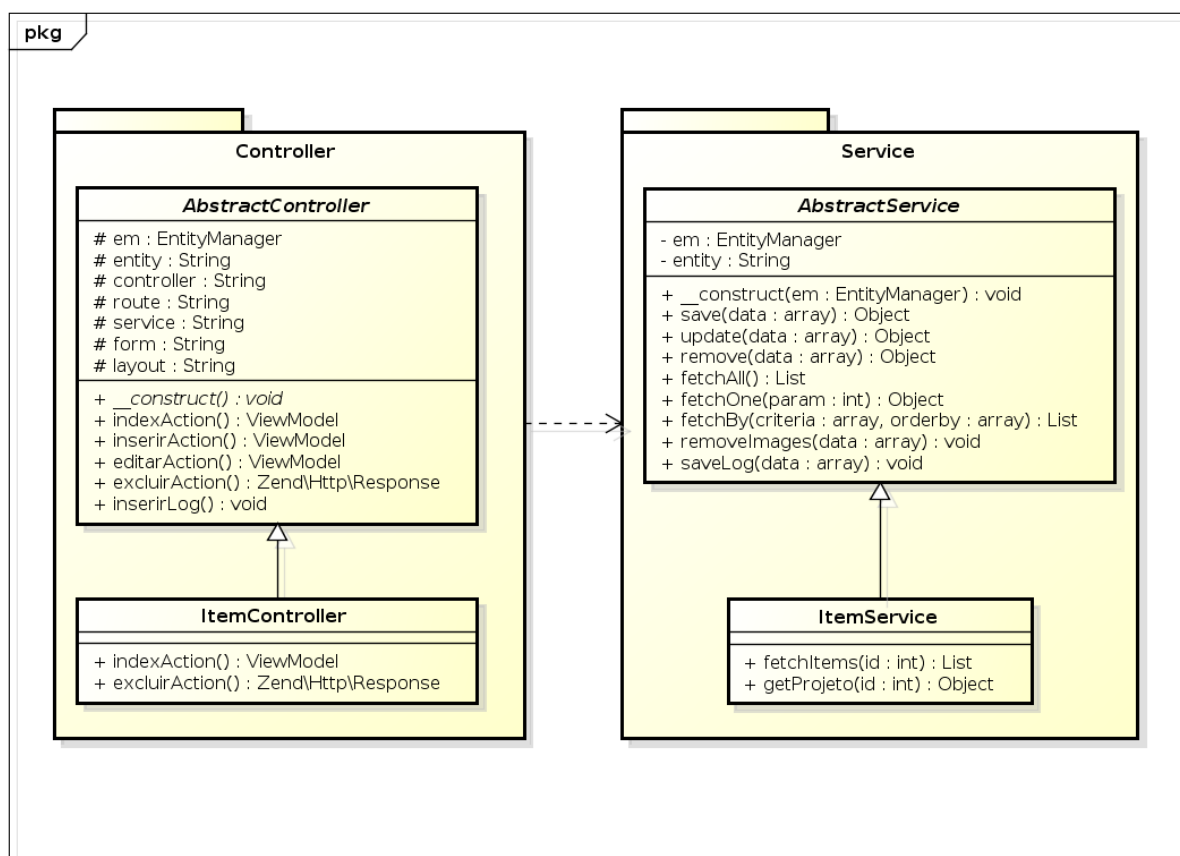


Figura 14: MER para o Caso de Uso Relacionar Material
 Fonte: Autoria própria

Para este Caso de Uso, foi gerada a *controller* *ItemController*, que estende a classe *AbstractController*, criada no módulo Base da aplicação. Sendo assim, *ItemController* herdará todos os métodos e atributos de *AbstractController*. Também foi implementada a classe *ItemService*, que estende a classe *AbstractService* que possui os métodos mais utilizados para manipulação de banco de dados. Pode-se observar esta representação no Diagrama de Classes da Figura 15.



powered by Astah

Figura 15: Diagrama de Classe - Controller e Model
 Fonte: Autoria própria

Segue uma breve descrição sobre as classes:

- **ItemController:** Esta classe possui dois métodos desenvolvidos especificamente para ela, que são:
 - **IndexAction:** Este método sobrescreve o *indexAction* da classe Pai. O método é sobrescrito, pois necessita de tratamento especial, uma vez que esta classe está listando os itens e também carregando um formulário de cadastro de itens.

- **ExcluirAction:** Este método sobrescreve o `excluirAction` da classe `Pai`. Precisa de um tratamento específico, pois foi preciso redirecionar para uma página específica.
- **ItemService:** Esta classe, além dos métodos herdados, ela também possui os seguintes métodos:
 - **FetchItems:** Faz a busca dos itens através do ID da tabela orçamento.
 - **GetProjeto:** Busca e retorna o projeto de um determinado orçamento através de seu ID.

Foi criado também uma classe dentro da pasta `Form` do módulo `Base`, nomeada como `ItemForm`. Nela possuem todos os campos que são necessários para inserir e editar um determinado item. Ela também realiza todas as validações necessárias nos campos e quando ocorre erros, informa ao usuário qual foi sua falha, basta ter o controle correto de erros na classe `ItemController`.

Cada função que retorna uma nova `ViewModel` deve conter seus arquivos de visualização, que se encontram dentro da pasta `View` do módulo. Esses arquivos devem ter o formato `PHTML` e ter somente o conteúdo referente a página, não sendo necessário a criação de um cabeçalho e um rodapé para o documento, pois existe uma configuração que adiciona um layout padrão para os arquivos.

Desta forma que os Casos de Uso foram desenvolvidos, cabe ressaltar que existem alguns arquivos de configuração padrões do `Zend Framework`, que devem ser adicionados para que alguns recursos funcionem.

3.5 VALIDAÇÃO DO SOFTWARE

A aplicação foi desenvolvida utilizando a metodologia `Scrum`, e a cada reunião das *Sprints*, também eram realizadas as *Sprints Reviews* com o próprio orientador, que representa o papel de `Scrum Master` e o aluno representando a Equipe de Desenvolvimento. As *Reviews* eram realizadas respondendo basicamente três questões sobre a *Sprint* que havia sido desenvolvida:

- Foi possível implementar todos os Casos de Uso adicionados na *Sprint*?

- O próprio orientador verificava se foram desenvolvidos todos os Casos de Uso propostos na reunião anterior, através de testes na aplicação e auxílio da tabela de acompanhamento das *Sprints* (Tabela 1).
- Os Casos de Uso desenvolvidos estão apresentando falhas?
 - O orientador e o aluno realizavam testes para ver se as funcionalidades implementadas apresentariam falhas. Se apresentassem, a correção era anotada para ser corrigida.
- Dos Casos de Uso implementados, o que poderia ser acrescentado?
 - No momento dos testes na aplicação, era também observado, implementações que estão funcionais, porém, poderiam ser melhoradas para acrescentar experiência do usuário.

Foi também realizado *Sprint Review* com *stakeholders*, alguns que não tinham conhecimento sobre gestão de fábricas de móveis planejados, e um marceneiro, que trabalha na empresa que está servindo de base para este projeto. Foram marcadas reuniões com o dono desta fábrica, porém, aconteceram imprevistos da parte da empresa e a *Sprint Review* não pode ser realizada.

Os *stakeholders*, que não tinham conhecimento sobre a área, deram um ótimo *feedback* sobre a aplicação, citaram somente algumas dicas de questões visuais sobre o sistema. As dicas foram analisadas e caso decidido que a alteração realmente melhoraria a experiência do usuário, a mudança seria implementada.

Já o Marceneiro, analisou a aplicação comentou que este projeto realmente iria ajudar no dia a dia da Fábrica, pois ajuda no controle de agenda, o que hoje é feito em uma planilha do Excel. Sugeriu também, que ocorresse notificação por e-mail toda vez que for agendado um evento. Essa sugestão foi muito bem aceita pela Equipe de Desenvolvimento, pois acarretava um melhoramento do sistema, logo, esta funcionalidade foi implementada.

3.6 MÉTRICAS

Ao final da implementação do projeto, foi utilizado um *plugin* do PHPStorm chamado PHP Metrics, uma ferramenta para análise de estatísticas para PHP

(PHPMetrics, 2015). Com essa ferramenta foi possível retirar informações analíticas sobre o projeto. Na Tabela 2, é apresentado um relatório com informações retiradas da ferramenta. Foi analisada somente a pasta Module do projeto, pois ela possui todas as classes que foram implementadas. Os arquivos visuais, Javascript e CSS, não entram nestes valores. Logo após a Tabela, está a descrição das informações.

Tabela 2: Métricas do projeto

NOME	VALOR
Linhas de códigos	7394
Linhas de códigos lógicos	1836
Classes abstratas	3
Classes concretas	61

Fonte: Autoria própria

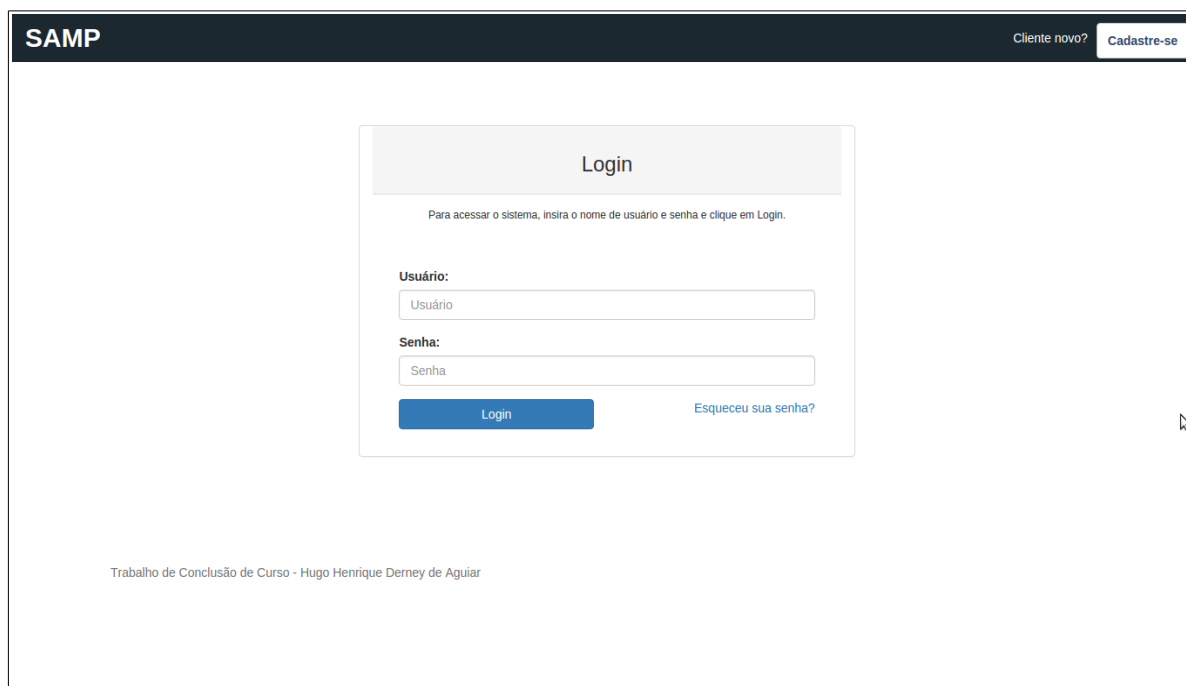
Nesta Tabela, são apresentadas informações sobre a quantidade aproximada de linhas de código da aplicação, a quantidade estimada de linhas de códigos lógicos da aplicação, que são representados pelas condições lógicas, e por fim, traz o número de classes abstratas e concretas que a aplicação possui.

4. SISTEMA SAMP

É uma aplicação desenvolvida para fornecer auxílio no gerenciamento de fábricas do segmento de móveis planejados. Trabalha com quatro tipos de usuários, o gerente, cliente, projetista e marceneiro. A seguir são descritas quais são as atividades de cada usuário e suas respectivas telas, em algumas poderá ser observado o acesso de um computador *Desktop* e também de um dispositivo móvel.

4.1 GERAL

Este subcapítulo mostra como é realizado o acesso ao sistema. Todos os usuários possuem a mesma tela de acesso, basta adicionar o nome de usuário e senha, definidos no momento do cadastro do usuário, conforme mostrado no Figura 16, acessado de um computador, e na Figura 17, de um dispositivo móvel.



A imagem mostra a interface de login do sistema SAMP em um navegador de desktop. No topo, há uma barra preta com o logotipo 'SAMP' à esquerda e os links 'Cliente novo?' e 'Cadastre-se' à direita. O formulário centralizado tem o título 'Login' e o texto instrutivo: 'Para acessar o sistema, insira o nome de usuário e senha e clique em Login.' Abaixo, há campos para 'Usuário:' e 'Senha:', cada um com um campo de entrada de texto. Um botão azul 'Login' está posicionado abaixo dos campos, e um link azul 'Esqueceu sua senha?' está à sua direita. No rodapé da página, há o texto: 'Trabalho de Conclusão de Curso - Hugo Henrique Derney de Aguiar'.

Figura 16: Tela de login – Desktop

Fonte: Autoria própria

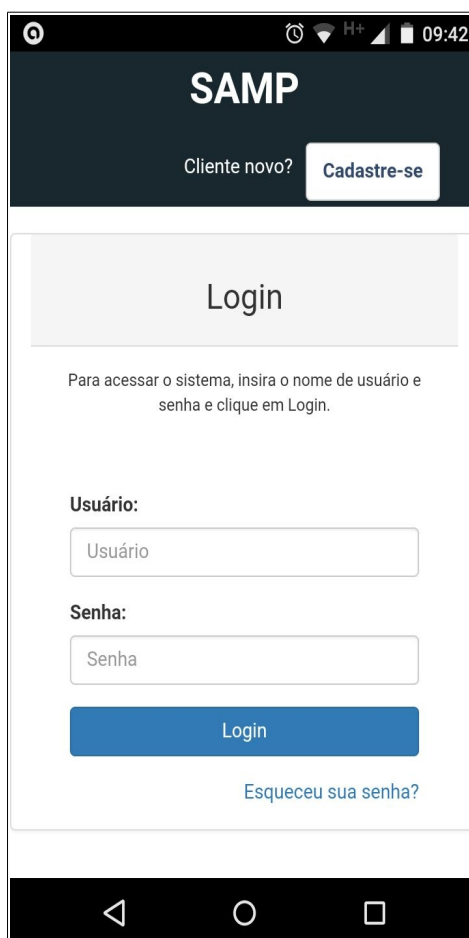


Figura 17: Tela de login – Dispositivo Móvel

Fonte: Autoria própria

4.2 VISÃO DO CLIENTE

Este capítulo traz as descrições das tarefas que o cliente poderá executar na aplicação.

4.2.1 Solicitar Orçamento

O cliente acessa a página de acesso ao sistema e clica no botão “Cadastrar-se” de cor branca, que se encontra no topo superior direito da página. Abrirá a página de cadastro de cliente, Figura 18 e 19, no qual o usuário terá que preencher as informações de cadastro obrigatórias, que estão marcadas com um asterisco (*) e poderá optar por preencher os campos opcionais. Após preencher as informações deve-se clicar no botão “cadastrar”.

SAMP Cliente novo? **Cadastre-se**

CADASTRAR-SE
Preencha os campos e clique em cadastrar.

Campos marcados com o símbolo * são obrigatórios

Nome: * **Email: ***

Username: * **Senha: *** **Telefone:**

Endereço:

Mapa Satélite

Google Dados cartográficos ©2015 Google | Termos de Uso

Cadastre-se

Figura 18: Tela de Cadastro de Cliente – Desktop

Fonte: Autoria própria

CADASTRAR-SE
Preencha os campos e clique em cadastrar.

Campos marcados com o símbolo * são obrigatórios

Nome: *

Email: *

Username: *

Senha: *

Telefone:

Endereço:

Figura 19: Tela de Cadastro de Cliente – Dispositivo Móvel

Fonte: Autoria própria

Após o cadastro, o usuário deverá realizar o login no sistema, conforme descrito no subcapítulo 4.1 deste documento. Feito isso, o usuário terá acesso à área de solicitação de orçamento, que pode ser acessada pelo menu lateral clicando em

“Orçamentos > Meus Pedidos”, nesta página, Figura 20, é possível visualizar o histórico de pedidos de orçamento e seu *status*, ou seja, se foi aceito, ou rejeitado, e também ao clicar sobre o pedido, a opção de cancelar.

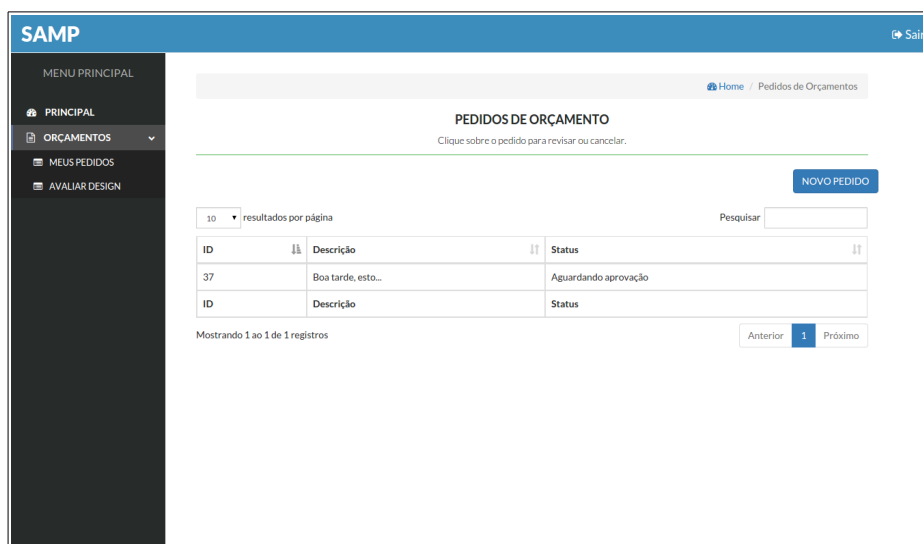


Figura 20: Tela de listagem de pedidos – Desktop

Fonte: Autoria própria

Para realizar um pedido de orçamento, basta clicar no botão “novo pedido”, que abrirá a página de cadastro de pedido, Figura 21. Para cadastrar um pedido é necessário preencher o campo “Descrição”. Também é possível adicionar fotos de móveis semelhantes ao desejado, para o projetista ter maior compreensão do gosto do cliente quando for desenvolver o projeto.

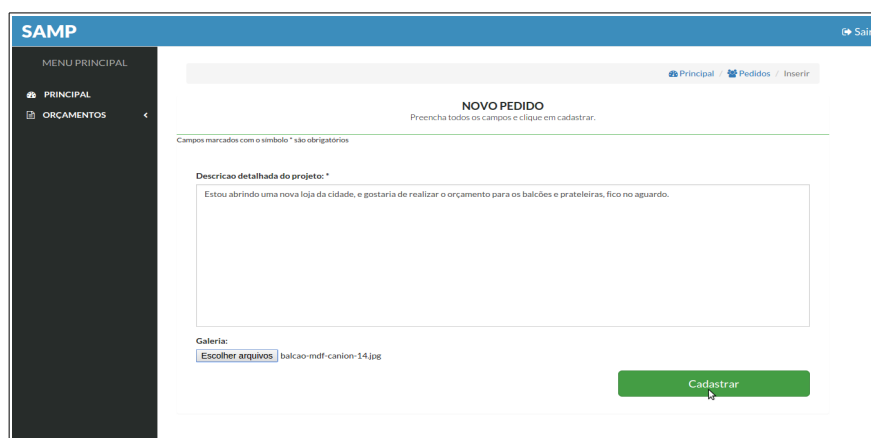


Figura 21: Tela de cadastro de pedido – Desktop

Fonte: Autoria própria

4.2.2 Avaliar *Design*

Assim que o pedido for aceito, e o projetista ter enviado a proposta de projeto, o cliente poderá ver o *design* clicando no menu lateral “Orçamentos > Avaliar *design*”, no qual ele poderá observar as imagens do projeto enviadas pelo projetista, Figura 22. Após analisar, ele poderá aceitar ou rejeitar o *design*.

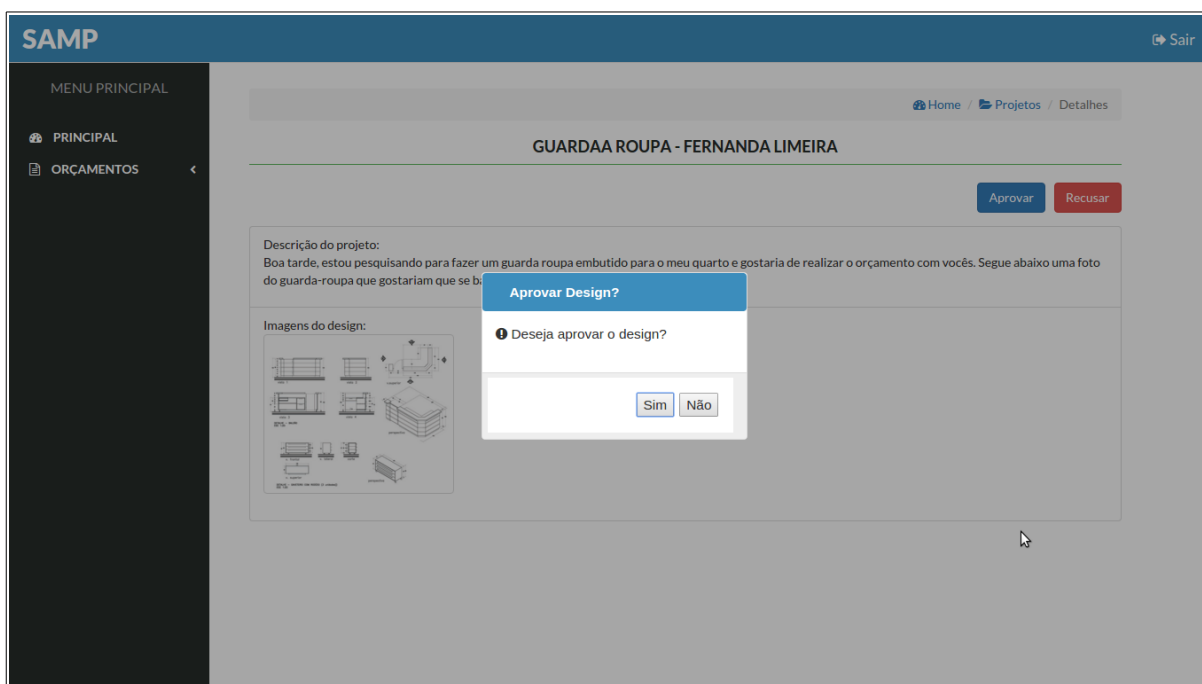


Figura 22: Avaliar *Design* – Desktop

Fonte: Autoria própria

4.3 VISÃO DO PROJETISTA

4.3.1 Realizar Design

O projetista obtém um calendário ao acessar o sistema, Figura 23 e 24, este tem como finalidade auxiliar o projetista e o gerente sobre visitas aos clientes com finalidade de tirar medidas do imóvel, para assim o projeto ser desenvolvido.

Clicando no item “Design”, localizado no menu lateral, é possível observar os projetos que foram destinados ao projetista e que estão com o desenvolvimento pendente, Figura 25. Clicando no botão “+ Detallhes” do projeto, o projetista poderá adicionar os arquivos do *Design*, Figura 26. Podem ser adicionados arquivos em

formato de imagem para ser mostrado para o cliente, ou arquivos que foram gerados pela ferramenta de *design*.

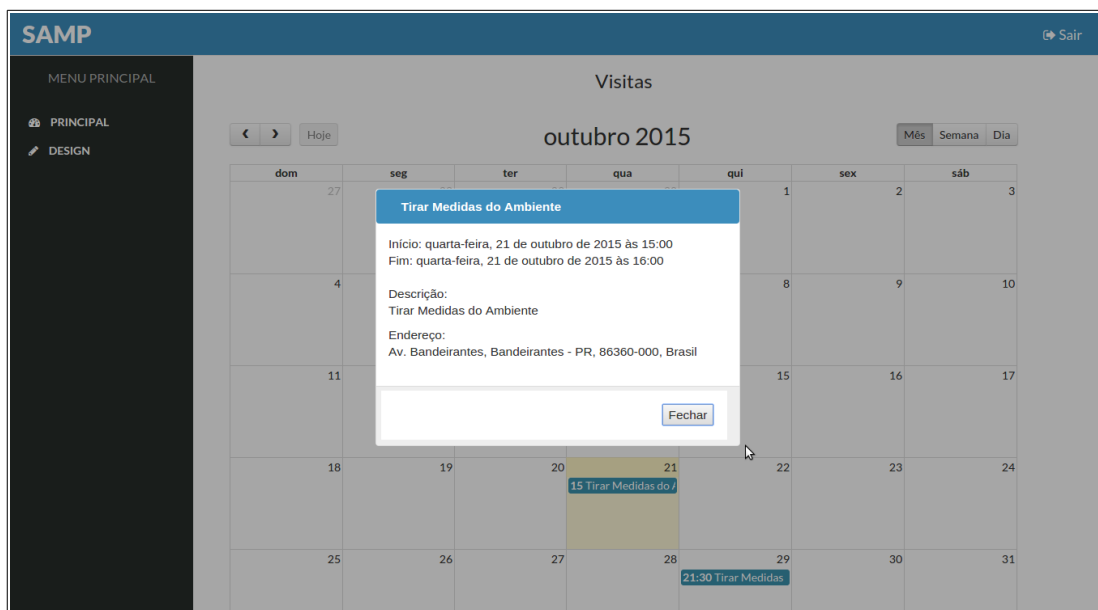


Figura 23: Agenda do Projetista – Desktop

Fonte: Autoria própria



Figura 24: Agenda do Projetista – Dispositivo Móvel

Fonte: Autoria própria

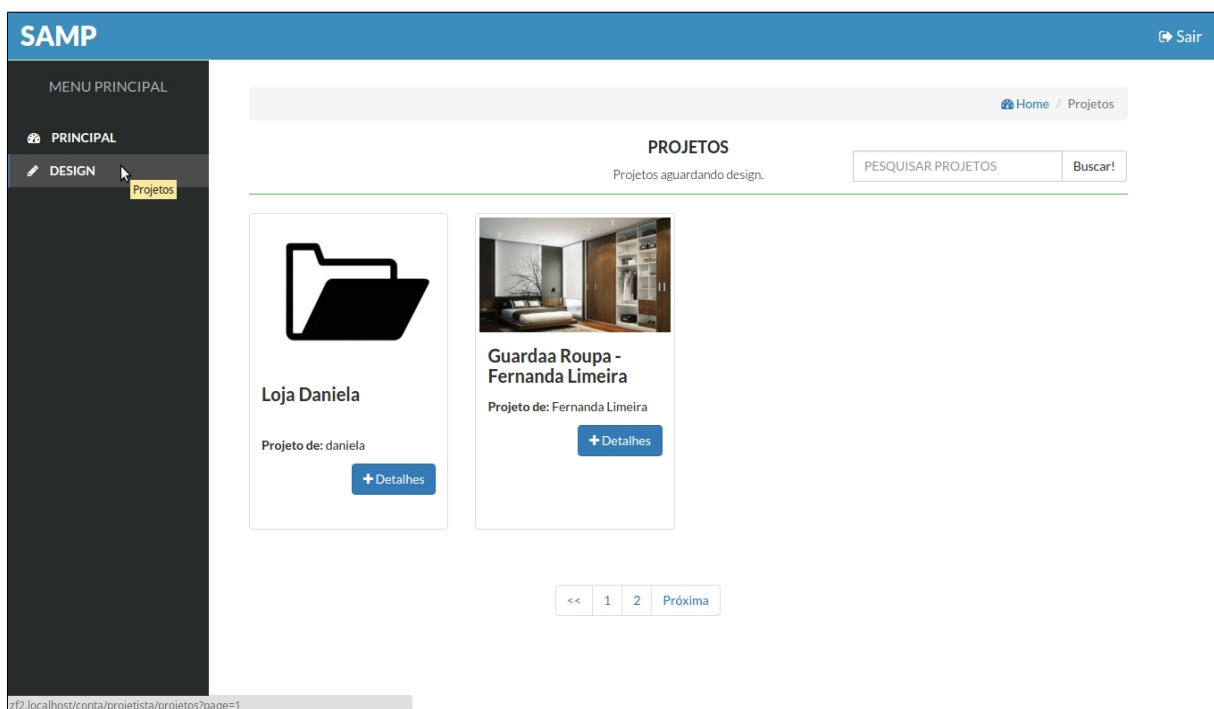


Figura 25: Listagem dos projetos – Desktop

Fonte: Autoria própria

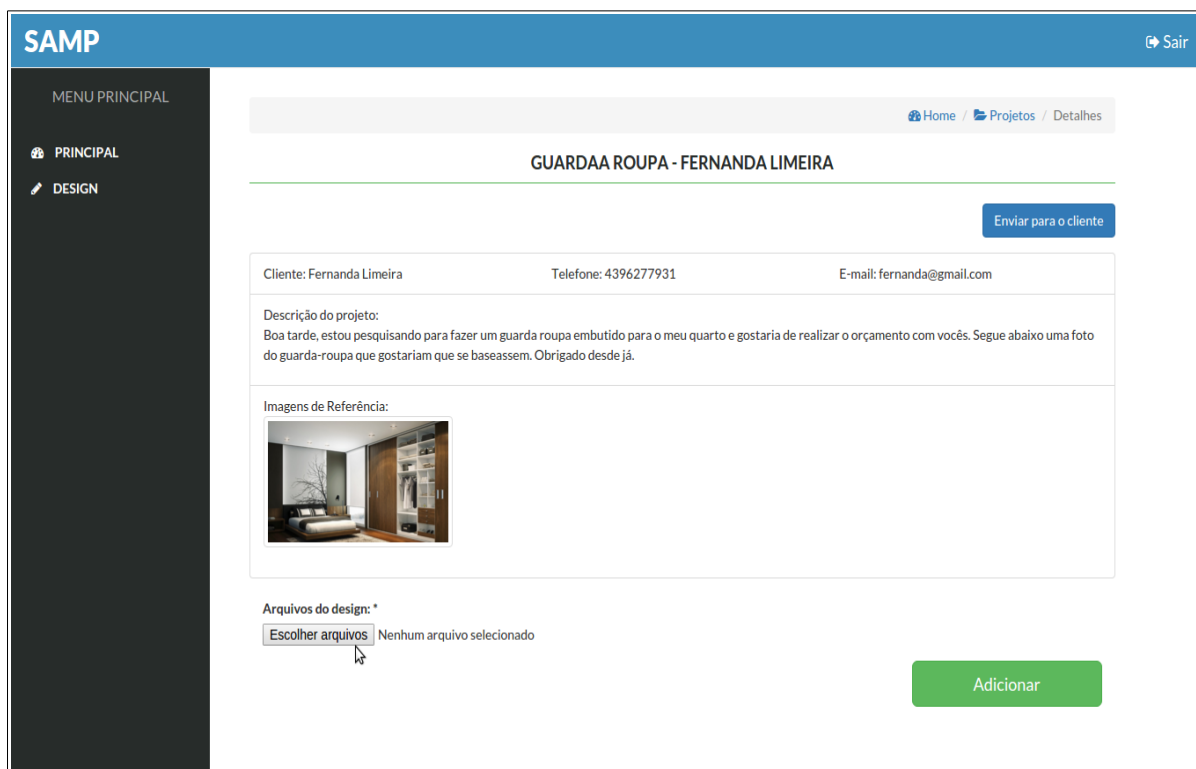


Figura 26: Adicionar arquivos do design – Desktop

Fonte: Autoria própria

4.3.2 Enviar para o cliente

Para enviar o projeto com o *design* adicionado para o cliente, basta clicar no botão “enviar para o cliente”, que está localizado ao acessar os detalhes do projeto, Figura 27.

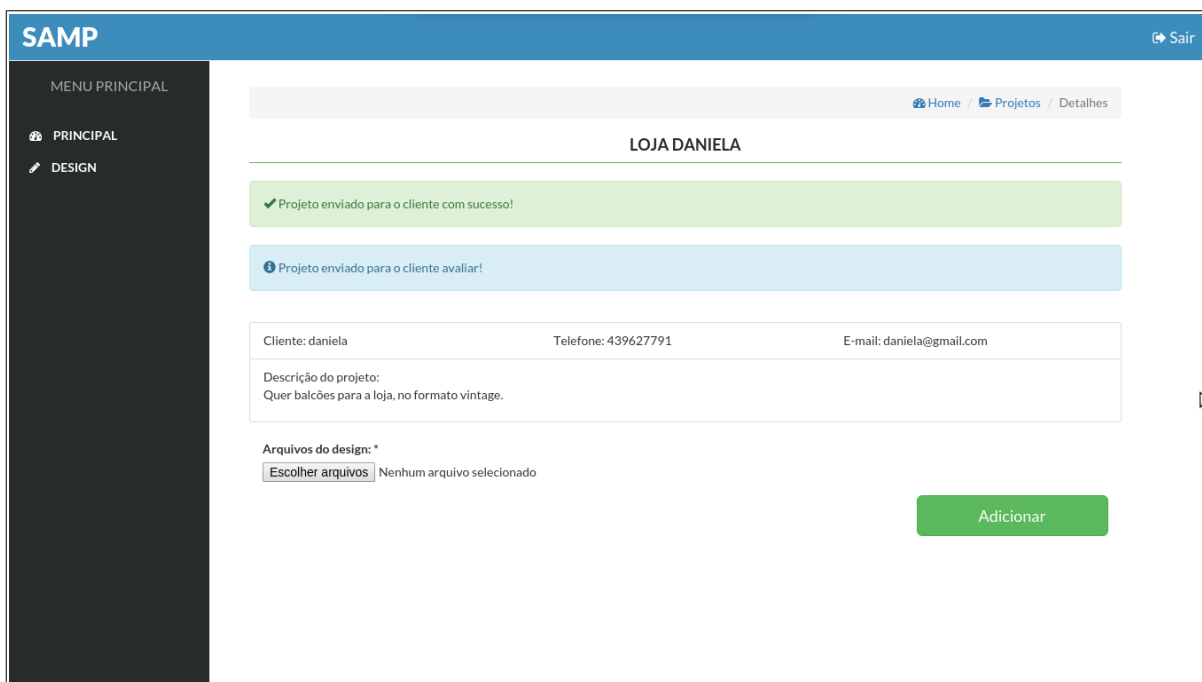


Figura 27: Projeto enviado para o cliente

Fonte: Autoria própria

4.4 VISÃO DO MARCENEIRO

4.4.1 Relacionar Material

Responsável por definir os materiais que serão utilizados no projeto, o marceneiro pode acessar esta atividade acessando o item “Projetos > Listar” no menu lateral, e na lista de projetos clicar no botão “+Detalhes”, nesta página, Figura 28 e 29, o Marceneiro conseguirá visualizar as informações necessárias do projeto para relacionar o material. Após estudar o projeto, basta clicar em “Relacionar Material” para inserir itens na lista, Figura 30.



Figura 28: Detalhes do projeto – Desktop

Fonte: Autoria própria

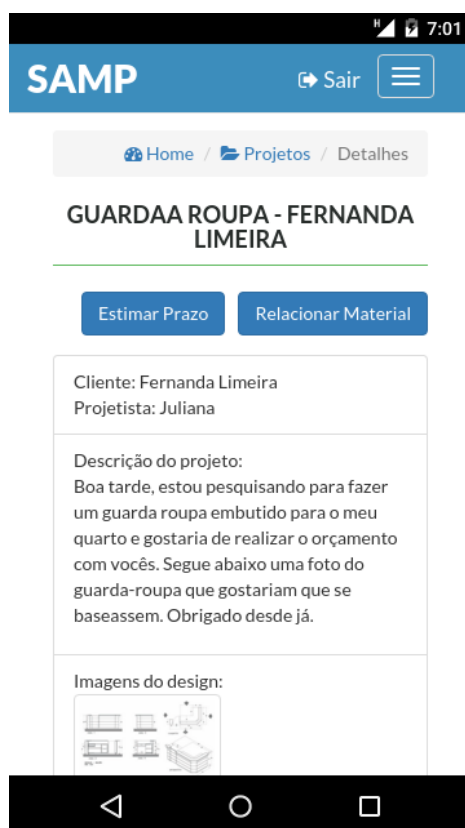


Figura 29: Detalhes do projeto – Dispositivo Móvel

Fonte: Autoria própria

The screenshot displays the 'RELAÇÃO DE MATERIAL' page in the SAMP application. The top navigation bar contains the logo 'SAMP' and a 'Sair' button. The left sidebar shows a 'MENU PRINCIPAL' with options for 'PRINCIPAL' and 'PROJETOS'. The main content area features a breadcrumb trail: 'Home / Guardaa Roupa - Fernanda Limeira / Items'. Below this, the title 'RELAÇÃO DE MATERIAL' is centered. The form includes three input fields labeled 'Qtid.', 'Nome:', and 'Observação:', followed by an 'Adicionar' button. A table below the form has the following structure:

Qtd	Nome	Observação	Excluir
Qtd	Nome	Observação	Excluir

Figura 30: Relacionar Material

Fonte: Autoria própria

4.5 VISÃO DO GERENTE

Este capítulo traz as descrições das tarefas que o gerente poderá executar na aplicação.

4.5.1 Realizar Agendamento

O Gerente consegue realizar agendamento de visitas para o projeto, acessando o menu lateral e clicando em "Projetos > Listar". Abrirá a página de listagem de todos os projetos, como pode-se observar na Figura 31 e 32.

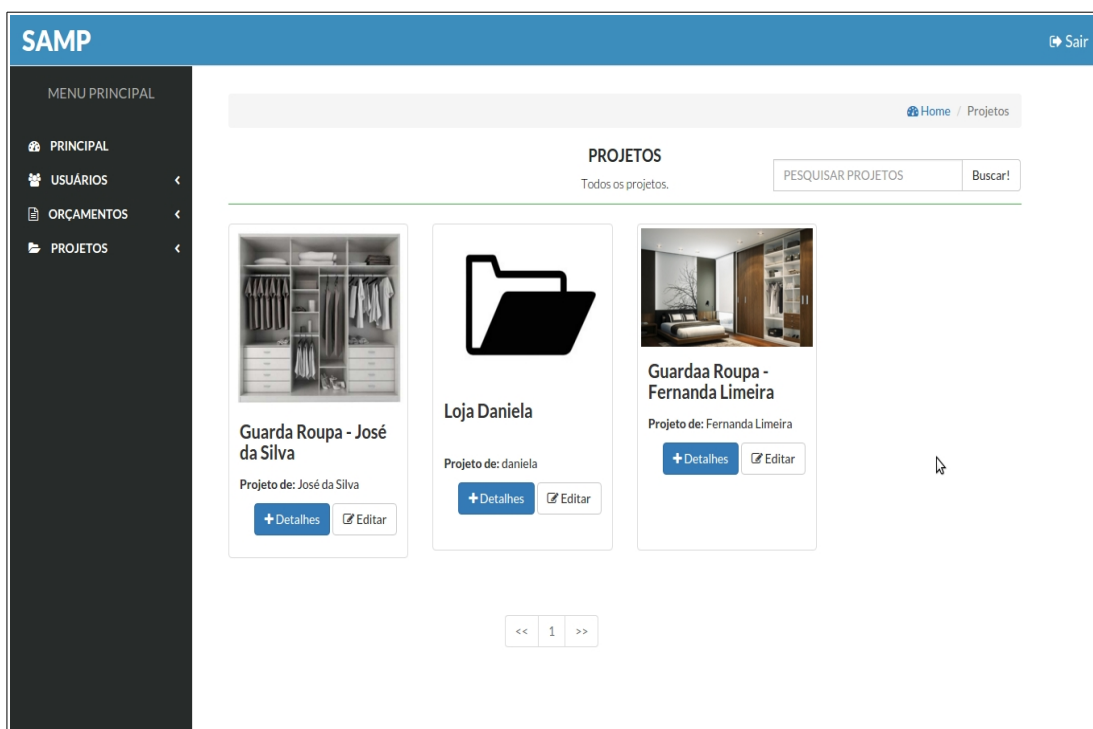


Figura 31: Todos os projetos – Desktop

Fonte: Autoria própria



Figura 32: Todos os projetos – Dispositivo Móvel

Fonte: Autoria própria

Para acessar a página de agendamento, é necessário clicar no botão “+Detalhes” para visualizar informações sobre o projeto, conforme a Figura 33.

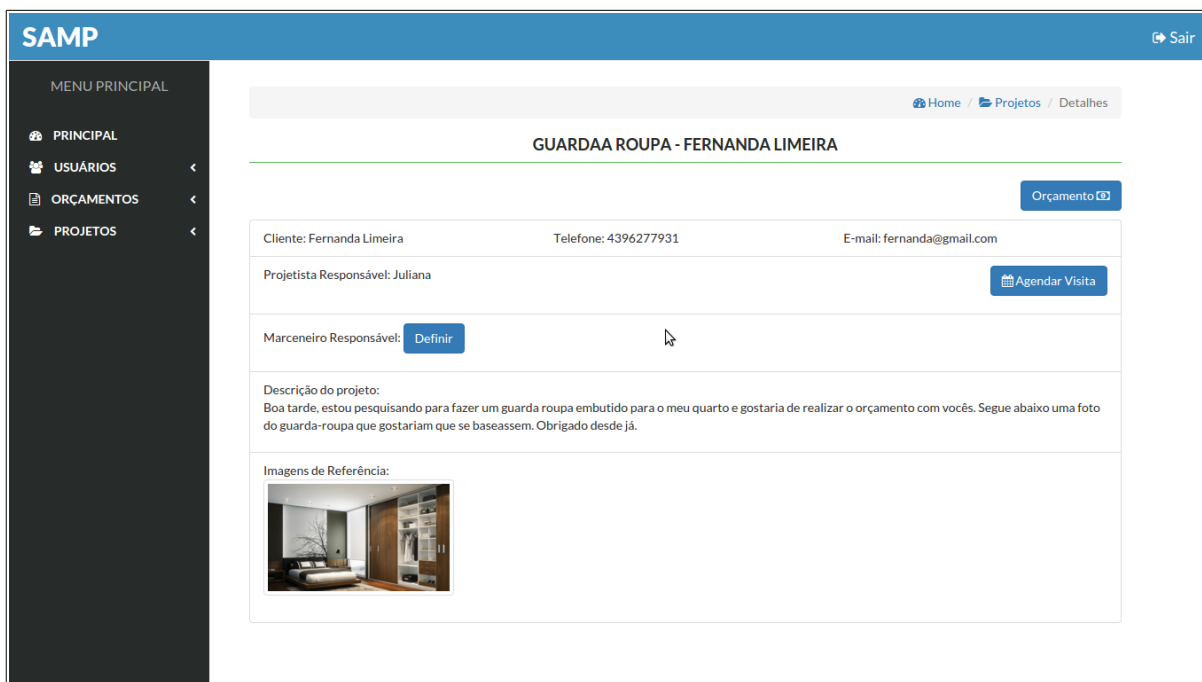


Figura 33: Visualizar detalhes do projeto

Fonte: Autoria própria

Se observarmos a Figura 33, é uma página com uma lista de informações sobre o projeto, entre elas, um botão para agendamento de visitas. Clicando neste botão, será redirecionado para a página de cadastro de visitas que já vincula a visita com o projeto que havia sido visualizado.

4.5.2 Escolher Marceneiro Responsável

Em todos os projetos, deve ter um marceneiro responsável, pois o orçamento depende muito das informações que o marceneiro fornecerá. Para definir o Marceneiro, basta acessar a página de detalhes do projeto, assim como na Figura 33, e clicar em “Definir”, na frente do título “Marceneiro responsável”. A página para adicionar um Marceneiro abrirá, assim como mostrado na Figura 34.

The screenshot displays the SAMP web application interface. At the top, there is a blue header with the SAMP logo on the left and a 'Sair' button on the right. Below the header is a dark sidebar menu with the following items: 'MENU PRINCIPAL', 'PRINCIPAL', 'USUÁRIOS', 'ORÇAMENTOS', and 'PROJETOS'. The main content area has a breadcrumb trail: 'Principal / Projetos / Adicionar Marceneiro Responsável'. The page title is 'DEFINIR MARCENEIRO RESPONSÁVEL' with the instruction 'Preencha todos os campos e clique em adicionar.' Below this, there is a note: 'Campos marcados com o símbolo * são obrigatórios'. The form contains a single required field labeled 'Marceneiro: *' with a dropdown menu showing 'Selecione um marceneiro'. A green 'Definir' button is positioned to the right of the dropdown.

Figura 34: Visualizar detalhes do projeto

Fonte: Autoria própria

Para adicionar o Marceneiro, basta escolher o nome através do campo, e em seguida, clicar em definir.

4.5.3 Calcular Preço Final

Responsável por auxiliar no cálculo do preço final do produto. Para isso, é preciso acessar a página de detalhes do projeto, conforme na Figura 33. A partir daí, é necessário clicar no botão "Orçamento", para acessar a página de resumo de valores. Nela, é possível ver qual são os itens necessários para desenvolver aquele projeto, o Gerente pode editar estes itens, adicionando o valor unitário dos mesmos. Também é mostrado o prazo de entrega que o Marceneiro estimou, pois é muito importante para o Gerente calcular o valor de mão de Obra. Relação de material e mão de obra são valores editáveis. A aplicação soma todas as informações de valores, e mostra o valor final, como apresentada na Figura 35.

Qttd.	Título	Obs.	Valor unt.	Valor total
2	Caixa de Parafuso		R\$ 120	R\$ 240
2	MDF		R\$ 170	R\$ 340
5	Fita		R\$ 135	R\$ 675
				Valor total: R\$ 1255

Prazo de entrega

Mão de obra
R\$ 1200

Valor total do Orçamento: 2455

Figura 35: Calcular preço final

Fonte: Autoria própria

4.5.4 Cadastrar Pedido de Orçamento

Além da possibilidade do cliente solicitar o pedido de orçamento através da Web, foi também adicionado suporte para o gerente cadastrar pedidos, pois os pedidos por telefone ainda serão em maior escala, pelo menos nos primeiros meses de implantação. Para adicionar um novo pedidos, basta acessar o menu lateral, e clicar em “Orçamento>Pedidos>Cadastrar”, que será mostrado um formulário para cadastro do pedido, conforme a figura 36.

NOVO PEDIDO

Preencha todos os campos e clique em cadastrar.

Campos marcados com o símbolo * são obrigatórios

Cliente: *

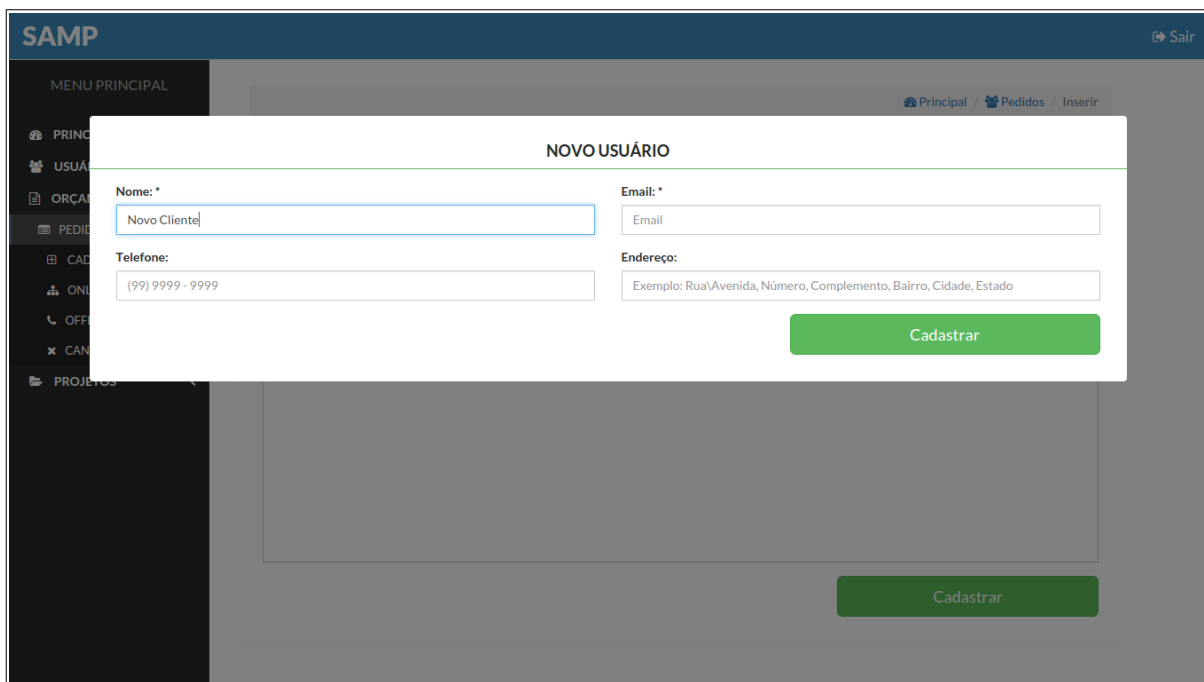
Selecione um cliente
Selecione um cliente
Daniela
Fernanda Limeira
José da Silva

Cadastrar

Figura 36: Cadastrar pedido – Gerente

Fonte: Autoria própria

Para adicionar um novo pedido, é necessário selecionar o cliente deste pedido, e caso o cliente não esteja cadastrado, é possível realizar um cadastro rápido clicando no ícone do lado do campo cliente, conforme na Figura 37. Após selecionar o cliente, basta adicionar uma breve descrição sobre o pedido e clicar em cadastrar.



A imagem mostra a interface de usuário do sistema SAMP. No topo, há o logotipo 'SAMP' e um link 'Sair'. Abaixo, um menu principal com opções como 'PRINCIPAL', 'USUÁRIOS', 'ORÇAMENTOS', 'PEDIDOS', 'CADASTRO', 'ONLINE', 'OFFLINE', 'CAMBIO' e 'PROJETOS'. O formulário 'NOVO USUÁRIO' está aberto, com os seguintes campos:

- Nome:** Campo com o texto 'Novo Cliente'.
- Email:** Campo com o texto 'Email'.
- Telefone:** Campo com o texto '(99) 9999 - 9999'.
- Endereço:** Campo com o texto 'Exemplo: Rua\Avenida, Número, Complemento, Bairro, Cidade, Estado'.

Dois botões verdes 'Cadastrar' são visíveis: um dentro do formulário e outro na parte inferior da tela.

Figura 37: Cadastro de cliente

Fonte: Autoria própria

5. CONSIDERAÇÕES FINAIS

5.1 LIMITAÇÕES E DIFICULDADES ENCONTRADAS

Neste trabalho, foi realizada a implementação do fluxo de atividade da empresa, em que alguns processos foram automatizados para auxiliar uma forma de controlar desde a entrada, até a saída do projeto da empresa.

Não foi desenvolvido o controle de estoque, somente o gerenciamento de pedido de material, a realização de orçamento e o acompanhamento do pedido.

A criação do *Design* do projeto também não terá relação com o software proposto, tendo em vista que existem ferramentas específicas para tal propósito, por exemplo, o AutoCAD.

No desenvolvimento e gerenciamento desta aplicação foram encontradas algumas dificuldades. Primeiramente, foi encontrada dificuldades em relação ao Framework utilizado. Este Framework tem como desvantagem o alto grau de dificuldade para dominá-lo, pois aplica conceitos de orientação a objetos e padrões de projeto, e se não há tanta experiência prática utilizando estes conceitos, a pessoa apresentará dificuldade de aprendizagem. Também pode-se considerar o fato de que sua documentação era toda na língua inglesa e o aluno não possuía conhecimento sobre esta ferramenta.

Houve também dificuldade nas etapas de *Sprint Review* da metodologia Scrum, o ideal seria realizar várias *Reviews* com a presença de *stakeholders*. Porém, foi solicitado várias vezes reuniões com os integrantes da marcenaria tomada como base. No entanto, a maioria não havia resposta e outras, as reuniões eram agendadas, mas não aconteciam por algum imprevisto da parte dos integrantes da empresa. Também devemos considerar que o aluno e a marcenaria se encontram em cidades diferentes.

5.2 TRABALHOS FUTUROS

Futuramente, o próximo passo seria trabalhar a implantação da aplicação na empresa, este trabalho envolveria treinamentos para garantir que as dúvidas dos

usuários sejam esclarecidas e também para detectar melhoras, que só poderão ser observadas conforme o tempo de uso.

Realizar testes de segurança da informação sobre a aplicação também seria o próximo passo, para garantir a integridade dos dados, pois, pela aplicação estar na Web, ela também corre mais riscos de sofrer ataques com intuito de roubar informações ou até mesmo de prejudicar o sistema.

Poderão ser implementadas futuramente questões que envolvem todo o controle financeiro da fábrica, que circundam o fluxo de caixa, contas a pagar, entre outras necessidades da empresa. Também o gerenciamento de estoque e inclusive uma maneira do cliente realizar o pagamento através do SAMP.

Esta aplicação foi desenvolvida tomando como base uma empresa, porém existe a possibilidade deste sistema ser utilizado por outras empresas no ramo. Desta forma, o SAMP só teria que sofrer algumas adaptações para empresas que possuem um fluxo de trabalho diferente, ou que desejam novas funcionalidades.

REFERÊNCIAS

IBGE, Instituto Brasileiro de Geografia e Estatística. **Acesso à Internet e posse de telefone móvel celular para uso pessoal**. Rio de Janeiro, 16/06/2013. Disponível em:

<<http://www.ibge.gov.br/home/presidencia/noticias/imprensa/ppts/00000012962305122013234016242127.pdf>>. Acesso em 15 de Setembro de 2014.

F/RADAR. **Panorama do Brasil na internet**. Outubro de 2013. Disponível em:

<<http://www.fnazca.com.br/index.php/2013/12/20/fradar-13%C2%AA-edicao>>.

Acesso em: 15 de Setembro de 2014.

SOUZA, Saulo Campos Nunes de.; Igarash, Wagner. **Web Design Responsivo no desenvolvimento de aplicações multi-dispositivos**. 2013, 13 f. Artigo (Departamento de Informática) – Universidade Estadual de Maringá (UEM), Maringá – Paraná, Brasil, 2013.

SCHAWABER, Ken; SUTHERLAND, Jeff. **Um guia definitivo para o Scrum: As regras do jogo**. 2013. 19 f. Guia da Scrum. Julho de 2013.

SALVI, Irene Lopes; PALMA, Jandira Guenka; DUARTE, Francisco Ricardo; FAVORETO, José Ricardo Favoreto. **Análise dos impactos de implantação de um sistema de informação na indústria moveleira**. 2005, 14 f. Revista Gestão Industrial, 2005.

GONÇALVES, Rodrigo Franco; GAVA, Vagner Luiz; PESSÔA, Marcelo Schneck de Paula; SPINOLA, Mauro de Mesquita. **Uma proposta de processo de produção de aplicações Web**. 2005, 14 f. Escola Politécnica da USP, 2005.

The PHP Group. **PHP: Hypertext Preprocessor**. 2001 – 2014. Disponível em: <php.net>. Acesso em: 27 de Setembro de 2014.

Zend Technologies Ltd. **Zend Framework**. Disponível em: <framework.zend.com>. Acesso em 19 de Agosto de 2015.

Oracle Corporation. **MySQL - The world's most popular open source database**. 2014. Disponível em: <mysql.com>. Acesso em 20 de Setembro de 2014.

The JQuery Fondation. **JQuery**. 2014. Disponível em <jquery.com>. Acesso em: 27 de Setembro de 2014.

W3C. **World Wide Web Consortium (W3C)**. 2015. Disponível em: <w3.org>. Acesso em 15 de Outubro de 2015.

W3C. **Cascading Style Sheets (CSS)**. 2015. Disponível em: <w3.org/Style/CSS>. Acesso em 15 de Outubro de 2015.

Doctrine Team. **Doctrine**. 2015. Disponível em <doctrine-project.org>. Acesso em: 22 de Outubro de 2015.

LÉPINE, Jean-François. **PHPMetrics**. 2015. Disponível em <phpmetrics.org>. Acesso em: 22 de Outubro de 2015.

Git. **Git**. 2015. Disponível em <git-scm.com>. Acesso em: 22 de Outubro de 2015.

Astah and Modeling Tools. 2005. Disponível em: <astah.net>. Acesso em 27 de Setembro de 2014.

Code licensed under MIT, documentation under CC BY 3.0. **Bootstrap**. Disponível em <getbootstrap.com>. Acesso em 19 de Agosto de 2015.

JetBrains s.r.o. **JetBrains PHPStorm**. Disponível em: <www.jetbrains.com/phpstorm/>. Acesso em 19 de Agosto de 2015.

LÉPINE, Jean François. **PhpMetrics**, Disponível em <phpmetrics.org>. Acesso em 19 de Outubro de 2015.

ADERMANN, Nils; BOGGIANO, Jordi. **Composer**. Disponível em: <getcomposer.org>. Acesso em 22 de Outubro de 2015.

GUEDES, Gilleanes T. A. **UML: Uma Abordagem Prática**. 2004, 319 f. Novatec Editora, 2004.

NIEDERAUER, Juliano. **PHP para quem conhece PHP**. 2008, 527 f. Novatec Editora, 2008.

CATLIN, Hampton; WEIZENBAUM, Natalie; EPPSTEIN, Chris. **SASS: Syntactically Awesome Style Sheets**. 2006 – 2014. Disponível em <sass-lang.com>. Acesso em 18 de novembro de 2014.

RUBY, Comunidade. **Linguagem de Programação Ruby**. Disponível em <ruby-lang.org>. Acesso em 18 de novembro de 2014.

Starling Software – Sistema para controle de fábricas de móveis personalizados. Disponível em <site.starling.com.br/marcenaria.php>. Acesso em 25 de novembro de 2015.

NC Móveis. Disponível em <www.ncsolucoes.com.br/NcMoveis-sistema-programa-software-de-gestao-de-loja-de-varejo-de-moveis.html>. Acesso em 25 de novembro de 2015.

BMS Marcenaria. Disponível em <bmssoftware.wordpress.com/2011/06/13/bms-marcenaria>. Acesso em 25 de novembro de 2015.