Plano Mobiliário - Sistema de Gestão Empresarial para o Segmento de Móveis Planejados

Mauricio Paz Pacheco[1], Rodrigo Souza D'Ávila[2], Regiane Orlovski[3]

[1] Tecnologia em Análise e Desenvolvimento de Sistemas

Faculdade Guairacá - Guarapuava – PR

mauriciopazdesign@gmail.com, rsdavila@gmail.com, regianeorlovski@hotmail.com

Abstract. *The objective of this work is to present the development of a software for a store in the furniture sector planned allowing a better control in manage, thus increasing the productivity of the store with more efficiency and organization, considering that nowadays management software is undoubtedly an essential tool for the competitive market. For development, was chosen the Java language, the Hibernate framework for modeling the database, among other tools. Thus, we obtained a system approved by the tenants interviewed, he said that the system will be able to meet the needs of the company.*

Resumo. *O objetivo deste trabalho é apresentar o desenvolvimento de um software para uma loja do setor de móveis planejados possibilitando, um melhor controle na gerencia, aumentando assim a produtividade da loja com mais eficiência e organização, tendo em vista que nos dias atuais um software de gerenciamento é sem dúvida uma ferramenta fundamental para o mercado competitivo. Para o desenvolvimento, foi escolhida a linguagem Java, o framework Hibernate para a modelagem do banco de dados, dentre outras ferramentas. Dessa maneira, foi obtido um sistema aprovado pelo lojista entrevistado, que segundo ele, o sistema será capaz de suprir as necessidades da empresa.*

Introdução

Um software de gestão empresarial, passou a ser uma ferramenta necessária para entrar na competitividade do mercado de trabalho atual. É comum de pequenas empresas não se utilizarem de uma aplicação para a gerencia de suas lojas, obtendo várias desvantagens perante a outras que se utilizam de aplicações para um melhor controle interno.

No setor de móveis planejados, estas aplicações são de suma importância, pois no dia a dia do trabalho, vários dados importantes são gerados com a necessidade de um fácil e rápido acesso a estas informações. Neste aspecto, foi desenvolvido um sistema que visa atender a estas necessidades.

O objetivo deste trabalho é apresentar o desenvolvimento de um sistema de gerenciamento do segmento de móveis planejados, onde foi empregado diversas paradigmas para a obtenção de um sistema robusto, útil e prático para o uso, possibilitando o armazenamento de dados importantes para a loja, como o cadastro de clientes, o gerenciamento de projetos da loja, sendo possível o inserção de dados de projetos para clientes, cadastro de ambientes para cada projeto, gerenciamento de

funcionários e seus respectivos cargos , possibilitando assim um gerenciamento mais ágil e prático para uma loja deste setor.

Dentre as ferramentas utilizadas para a implementação do software, estão o *NetBeans IDE*, utilizado como ambiente de desenvolvimento, o *MySQL WorkBench* usado para a visualização do Banco de Dados criado pelo Framework *Hibernate*, que tem como tarefa a criação do Banco de Dados dentro do próprio código do software, o *iReport*, usado para a criação de relatórios ricos em detalhes, e a linguagem de programação Orientada a Objetos Java.

Fundamentação Teórica

Segundo o BNDES (2013), o setor de móveis planejados no Brasil, possui predominância de pequenas e médias empresas que atuam em um mercado bastante segmentado. A demanda de móveis, varia de acordo com o nível de renda da população. Vários fatores influenciam essa demanda, que são as mudanças no estilo de vida da população, os aspectos culturais, o investimento em *marketing,* entre outros.

Ainda o BNDES (2013), cita que a mais nova tendência entre os consumidores de classe pequena e média são os móveis modulares. Produzidos em módulos adaptáveis a um determinado projeto, esses móveis, cuja demanda vem crescendo muito no Brasil, reúnem qualidade e funcionalidade, a um custo reduzido, eles permitem que o cliente aproveite melhor o espaço físico disponível, adquirindo o produto em módulos pré-montados.

De acordo com o Jornal Hoje (2013), o bom desempenho da economia e o aumento da renda permitiram que as famílias de baixa renda pudessem realizar seus sonhos com mais facilidade, por conta disto, o setor de móveis planejados cresceu junto com o mercado. Com a utilização de novas tecnologias, principalmente em ferramentas de corte e design, os móveis planejados estão conquistando o mercado brasileiro.

Porém, estudos divulgados pelo SEBRAE (2013) mostram que de 100 empreendimentos criados, 73 sobrevivem aos primeiros dois anos de atividade. Dentre alguns fatores que levam uma empresa a fechar as portas neste período de tempo, um dos principais motivos está na escassez de administração do negócio, ficando clara a importância de softwares empresariais, que atuam exatamente neste tipo de problema.

Para o desenvolvimento, foi escolhida o paradigma que acompanhará desde o início até a conclusão da implementação do *software* em questão. O paradigma escolhido foi o ciclo de vida em cascata. Segundo Pressman (2011), sugere uma abordagem sequencial e sistemática para o desenvolvimento de um sistema. O modelo cascata, basicamente passa por todos as fases de desenvolvimento sequencialmente, e não passa para a próxima fase, sem que a primeira esteja concluída. Se inicia com o levantamento de necessidades por parte do cliente, avançando pelas fases de planejamento, modelagem, construção, emprego e culminando no suporte contínuo do software concluído.

Segundo Ian (2011), o processo se inicia com a definição do que o sistema será capaz de fazer, definindo funções juntamente com suas prioridades. Em seguida, se faz o projeto do software, onde é baseado na definição das funcionalidades do sistema onde foram definidas na etapa anterior. O modelo em cascata deve ser usado apenas quando os requisitos das etapas anteriores estiverem bem compreendidos e pouco provavelmente venham a ser mudados durante ao desenvolvimento do sistema.

Outro modelo adotado para o desenvolvimento foram os diagramas *UML*. A *Unified Modeling Language /* Linguagem Unificada de Modelagem *(UML),* que é uma linguagem padrão para modelagem orientada a objetos, e tem como papel auxiliar e visualizar o desenho e a comunicação entre objetos de um sistema.

Melo (2010), cita que hoje é indiscutível que a UML seja um padrão de mercado para a modelagem de sistemas orientado a objetos, sendo uma linguagem para especificação, construção visualização e documentação de um futuro software. O grande objetivo da UML, é obter uma maior visão do sistema antes de começar a escrever os códigos, eliminando assim possíveis erros e acelerando o processo de desenvolvimento.

Dentro dos diagramas, há o diagrama de caso de uso. Segundo Melo (2010), descreve uma sequência de ações que representam um cenário principal e cenários alternativos, com o objetivo de demonstrar o comportamento de um sistema, ou seja, após feito o levantamento de requisitos, é feita uma documentação para o entendimento não somente dos desenvolvedores, e sim para toda a equipe de desenvolvimento, com o objetivo de facilitar o trabalho evitando que informações importantes não se percam. Melo (2010), ainda cita que utilizando a modelagem de casos de uso, o primeiro passo do desenvolvedor e separar as funcionalidades do sistema, após este passo deve-se agrupar um conjunto de ações com um objetivo bem definido.

O padrão *Model View Controller /* Modelo Visualização Controlador *(*MVC*)* foi adotado para o desenvolvimento do sistema. Para Gabardo (2012), basicamente o MVC separa as camadas lógicas e negócio das camadas de apresentação, ou seja, padroniza o sistema e separa os devidos arquivos de acordo com suas funções, tornando o código mais organizado e fácil de manusear.

Segundo Gabardo (2012) é um padrão de projeto adotado para separar as camadas lógicas das regras de negócio da camada de apresentação de um sistema. Nas três camadas distintas, o *Model* trabalha com a fonte dos dados, o *Controller* é responsável pela comunicação entre as camadas de dados do *Model* e da camada *View* que é a principal camada de apresentação sendo nesta camada que os dados entram e saem para serem exibidas para o usuário.

Entretanto, uma interface deve ser intuitiva, simples e fácil de usar. Com estas qualidades, faz com o que os usuários se sintam confiantes e satisfeitos por atingirem seus objetivos com muito menos esforço, com menos tempo, e menos erros. Contudo, interfaces que dificultam o uso do sistema, trazem stress ao usuário, frustração e também perda de desempenho nas tarefas com o mal uso do software. Cybis (2010).

Segundo Cybis (2010), depende do posicionamento, da ordenação e da forma dos objetos para que o usuário tenha uma melhor compreensão de uma tela do sistema. Os usuários detectarão os itens e componentes, e compreenderão suas relações mais facilmente se estes componentes e itens forem colocados na tela com mais precisão, relação e se forem organizados de maneira organizada, como em ordem alfabética, frequência, por data, formas de uso, etc. Entretanto, é necessário compreender a função que uma determinada tela possui para que os componentes, imagens e textos sejam colocados da maneira com que o usuário veja a tela, e identifique suas similaridades ou diferenças.

O software foi escrito com a linguagem de programação Java, que é orientada a objetos utilizada para o desenvolvimento de aplicações, possibilitando sua execução em

diferentes Sistemas Operacionais. Dentre as diferentes áreas que a linguagem Java está organizada, a aplicação desenvolvida utilizou a *Java 2 Standard Edition*, provê um ambiente para aplicações em *desktop,* Somera (2011).

Já para Arnold, Goslin e Holmes (2007), a linguagem Java compartilha muitas características comuns da maioria das linguagens atualmente em uso. Ela foi projetada com base em outras, mas principalmente com C e C++. Por ser Orientada a Objetos, Arnold, Goslin e Holmes (2007), completam que a unidade fundamental de programação da linguagem de programação Java, são as classes, que fornecem a estrutura para os objetos e os mecanismos para fabricar a partir de uma definição. Classes definem métodos, que são escopos de código executável que são o foco da computação e que manipulam os dados armazenados em objetos, Arnold, Goslin e Holmes (2007).

Santos (2003), cita que a Programação Orientada a Objetos, é um paradigma de programação onde o código é separado em classes e objetos, criados para representar e processar dados usando *softwares*. O Java por ser uma linguagem Orientada a Objetos, proporciona ao sistema criado, a possibilidade de ser modular e flexível, além de reutilizar códigos já criados diminuindo a redundância e facilitando uma possível manutenção do sistema. Segundo Silva (2011), a orientação a objetos se preocupa com a reutilização de código, redução de erros e do tempo de manutenção.

De acordo com Silva (2011), com a crescente importância de softwares, muitas empresas estão aderindo ao paradigma da Orientação a Objetos, adotando o uso de diversas ferramentas para um bom crescimento do sistema, pois devido ao grande crescimento da *web*, fica claro a necessidade do bom uso de técnicas disponíveis para um melhor desenvolvimento de um sistema robusto e eficaz. Silva (2011) cita também sobre a importância dos modelos antes de se iniciar a programação em si, pois segundo ele, a modelagem de um sistema existe unicamente porque não é possível compreender os sistemas completamente antes de sua implementação, tornando a modelagem muito importante para um desenvolvimento eficaz.

Outro fator importante para a integridade de um sistema é o Banco de Dados. Segundo *OnJava* (2005)*,* a maioria das aplicações necessitam de trabalhar com dados. Softwares escritos em Java, durante a execução trabalham com estes dados, mas assim que a aplicação se fecha, os dados se perdem, necessitando assim de uma maneira de armazena-los e representa-los como objetos.

Sendo assim, a aplicação se utilizará do *MySQL,* que segundo a documentação *MySQL*(2013), um Banco de Dados relacional armazena os dados separado em tabelas, e não em um único lugar. O *MySQL,* oferece o modelo lógico com objetos, tabelas, visualizações, linhas e colunas com um ambiente flexível, possibilitando carregar diversas regras que reagem a ligações entre diferentes tabelas e diferentes campos de dados. Segundo a documentação do *MySQL*(2013)*,* o *SQL* do nome *MySQL, "Structured Query Language",* é uma linguagem padrão mais usada para a persistência em Banco de Dados. Entretanto, escrever códigos *SQL* manualmente para executar tarefas pode ser mais trabalhoso. Pensando nisso, existem ferramentas para acelerar o processo de persistência em Banco de Dados, e uma delas é o *framework Hibernate*, que foi utilizado no desenvolvimento do sistema.

Segundo a documentação do *framework Hibernate(2013)*, trabalhar com um software orientado a objetos e Banco de Dados relacional pode ser trabalhoso e demorado. Muitos desenvolvedores e arquitetos de software estimam que é necessário até 30% do

seu tempo para lidar com este problema de infraestrutura. O *Hibernate* aborda diretamente este desafio, fornecendo a capacidade de mapear a representação de dados de um modelo de objeto, para um modelo de dados relacional e seu esquema de banco de dados correspondente

Basicamente, o *Hibernate* faz o mapeamento das classes Java para tabelas do banco de dados, e tipos de dados Java para tipos de dados SQL. Segundo a documentação do *Hibernate* (2013)*,* o objetivo do framework é aliviar o desenvolvedor de 95% das tarefas de programação relacionadas com a persistência de dados comuns, eliminando a necessidade de um processamento de dados feito à mão usando SQL e JDBC.

Para auxilio e visualização do banco de dados criado, foi utilizado também o software *MySQL Workbench.* O *MySQL Workbench,* fornece uma ferramenta gráfica para trabalhar com o banco de dados. Com ele, foi possível visualizar o que estava sendo criado com o *Framework Hibernate,* facilitando a modelagem do banco.

Para escrever os códigos, é necessário muito mais do que um simples editor de texto, é necessário que o editor recue linhas, associe palavras e colchetes, dentre outras funções. Portando para o desenvolvimento, foi utilizado a IDE *NetBeans 7.3*. Segundo o próprio site do software *NetBeans* (2013), o *NetBeans* é um ambiente de desenvolvimento que pode ser usado para desenvolver aplicativos desktops profissionais independentes de Sistemas Operacionais, oferecendo um excelente suporte para as tecnologias e aprimoramentos Java mais recentes. Com ele, é possível criar interfaces de um sistema desktop de maneira muito fácil e rápida simplesmente arrastando componentes em uma área gráfica.

Tratando-se ainda de IDEs, para a criação de relatórios que são gerados no sistema, foi utilizado o *iReport.* Através de uma interface gráfica e rica, o *iReport* oferece várias funções muito importantes para a geração de relatórios em pouco tempo. Segundo o *iReport-Tutorial* (2013), O iReport*,* é um design visual para *JasperReports,* que é um mecanismo de relatório de código aberto para a comunidade Java. Com o *iReport,* é possível fazer relatórios sem a necessidade de conhecer a sintaxe XML.

Desenvolvimento

O primeiro passo antes do desenvolvimento, foi o levantamento de requisitos, onde de início, foi realizada uma conversa informal com um dono de uma loja de móveis planejados. Neste momento, foi descrito a rotina de trabalho dos funcionários da loja juntamente com as diversas necessidades que a loja possui na gerencia das informações que ele obtinha ao longo do dia-a-dia com seus clientes.

Nestas informações repassadas pelo lojista, foi informado a rotina que se tem no decorrer do trabalho antes de fechar algum negócio, foi dito que em uma loja deste setor, há um ciclo de vida durante o processo de venda de móveis, que segundo o lojista, a dificuldade na gerencia deste ciclo, é trabalhoso.

Segundo o dono da loja, o ciclo de vida, funciona da seguinte forma: O cliente vai até a loja, e solicita um orçamento, então é marcado um horário com o cliente para que um projetista possa ir até o local onde o cliente deseja fazer os móveis, então as medidas do ambiente são escritas no papel e levadas para a loja novamente para o desenvolvimento do projeto. Após este passo, entra a vez do projetista, que cria o projeto e marca

novamente outro horário com o cliente para que ele possa retornar até a loja para ver o projeto e realizar a compra.

Portanto, neste processo longo, foi visto a necessidade de armazenar todas as informações coletadas, para um melhor controle deste ciclo de vida da venda, desde os dados do cliente, o armazenamento correto das medidas que foram tiradas no ambiente, as observações que o cliente impôs para o móvel, dentre outras informações importantes, que, segundo o dono da loja, irão influenciar na decisão do cliente para fechar o negócio, de acordo com o bom e rápido atendimento.

Na loja do empreendedor entrevistado, foi visto que todo este processo, sempre foi feito utilizando somente anotações em agendas com papel e caneta, obtendo assim diversas dificuldades desde o acesso a todas estas informações, até a grande dificuldade que se tinha para controlar todo o processo do ciclo de vida de vários projetos ao mesmo tempo de seus clientes.

Após a conversa informal, foram coletados amostras de documentos e anotações reais feitas que foram usadas em cada processo no decorrer do ciclo de trabalho do lojista e dos funcionários. Com estes documentos, foi certificado a necessidade do sistema focar neste ciclo de vida dos projetos, que seriam o cadastro de clientes físicos e jurídicos, cadastro dos projetos, referente aos clientes, cadastro de ambientes para cada projeto separadamente, gerando relatórios rápidos com o detalhamento dos projetos.

Outra função importante não solicitado pelo lojista mas que é de suma importância para o controle dos dado, foi a necessidade do cadastro de mais de um endereço por cliente, pois segundo documentos e anotações reais da loja, há vários casos em que o cliente solicita um projeto para uma casa que ainda está em construção, ou então o cliente possui algum comércio, dentre outros casos parecidos, trazendo assim a necessidade de se armazenar estes endereços que serão úteis para ocasiões como: Entregar notas fiscais, pagamentos, dentre outras situações importantes, que na tela de cadastro de endereços, estes dados poderão ser inseridos em um campo chamado Referência, facilitando a percepção do usuário referente ao endereço.

Visto que a maior dificuldade encontrada pelo dono da loja de móveis planejados é o controle do ciclo de vida dos projetos de seus clientes, foi incluída e aprovada pelo lojista, a opção de incluir no cadastro de cada projeto, uma opção referente ao estado que o projeto se encontra, juntamente com um filtro dos projetos. Esta funcionalidade visa trazer total controle dos projetos da loja, com uma organização completa da listagem dos projetos, fazendo com que cada funcionário dê as devidas prioridades para cada cliente. Nesta função, foi criado uma janela unicamente para a exibição destes projetos de acordo com o seu estado no ciclo de vida.

Portanto, nesta função do sistema, foi criado uma única janela com uma tabela para cada estado do ciclo, sendo assim, seria mais rápido e pratico ver os projetos sem precisar fazer o filtro manualmente, portanto, foi criado uma classe chamada, *CiclosDeProjetos.java*, onde será exibido os projetos e filtrados automaticamente de acordo com a evolução do ciclo.

Nesta coleta de amostras de documentos, foi visto que a loja utiliza um software de modelagem 3D para a realização de projetos. O software utilizado gera um arquivo para cada projeto, e em alguns casos, é gerado imagens de cada ambiente para um projeto. Como padrão, os funcionários adotaram uma forma de organização dos arquivos referente

aos clientes, que dentro destas pastas, são armazenados arquivos 3D referente ao projeto de um cliente, e algumas imagens do projeto. Porém, esta forma de organização não é ágil o suficiente, pois dentre tantos clientes, várias pastas são gerados mensalmente com diversos arquivos, trazendo dificuldades e redundância de nomes, dificultando a procura por estas pastas. Segundo o lojista, estas pastas são usadas com bastante frequência, e são usadas também como uma listagens dos clientes que a loja possui. A Figura 1. é uma imagem real que apresenta uma pequena arquitetura de pastas com o padrão adotado pela loja.

Figura 1: Imagem referente a documentação levantada para o levantamento de requisitos.

A Figura 1, deixa clara a redundância de nomes e dificuldade no filtro destas pastas, pois não há uma padronização nos nomes, contudo se esta arquitetura de pastas adotado pela loja é usada com bastante frequência, fica clara a grande dificuldade no trabalho do dia. Neste mesmo problema, segundo o lojista, há casos em que o cliente vai até a loja pra ver um determinado projeto que solicitou de uma forma mais rápida, contudo o software utilizado para criar os projetos é um software bastante robusto e demora para

abrir os arquivos, isto segundo o lojista seria um problema que deveria ser solucionado de alguma forma.

Para solucionar o problema de redundância e filtro das pastas, e agilizar este processo de pesquisa e visualização rápida de um projeto realizado, o software implementado, pode armazenar imagens referentes a cada ambiente de cada projeto. O usuário simplesmente lista os projetos que o cliente cadastrado possui, e logo será listado seguido dos projetos os ambientes referentes ao mesmo, agilizando assim a pesquisa, e possibilitando visualizar uma imagem referente ao projeto e ambiente do cliente, eliminando a necessidade de abrir os arquivos em formato do software 3D, que gerava lentidão em uma rápida apresentação do projeto para o cliente.

Para agilizar ainda mais este processo, será possível abrir os arquivos dos projetos em 3D com um único clique em um botão no próprio software, sem ter a necessidade do usuário abrir o *Explorer* do sistema operacional para procurar este projeto, focando o usuário a interagir ainda mais com o sistema, trazendo mais utilidade, e eliminando a rotina que os funcionários tinham de buscar todos os arquivos pelo *Explorer* do Sistema Operacional.

Também no cadastro de ambientes, foi vista a necessidade de armazenar as medidas que são anotadas pelo projetista no ambiente solicitado pelo cliente. Segundo o projetista e segundo as documentações dadas pela loja, as medidas são feitas com papel e caneta. Segundo ele, são anotados informações como medidas das paredes, pontos de água, gás, tomadas, etc. Durante a documentação foi visto o grande volume de papeis que se tinha destas anotações, que muitas vezes, era necessário procurar um único papel de um cliente no meio de tantos.

Buscando a solução para este problema, foi incluído no sistema, o cadastro de imagens referentes as medidas anotadas na casa do cliente, portanto, o usuário digitalizaria as medidas e faria a inserção desta imagem no cadastro do ambiente, sendo assim, à qualquer momento, o usuário poderia imprimir novamente as medidas quantas vezes, e quantos folhas fossem necessárias, e os papeis que já foram digitalizados, poderiam ser descartados, trazendo mais organização tanto na loja fisicamente, como no controle dos dados lógicos.

Foi vista também, a necessidade de cadastrar os funcionários da loja, sendo possível informar o *status* do funcionário, informando ao usuário se ele ainda trabalha, ou deixou de trabalhar na empresa. Dentro do cadastro de funcionários, está disponível o cadastro da foto da pessoa, facilitando a rápida visualização e deixando o software mais agradável para a utilização. Nesta mesma funcionalidade, está disponível também, a opção da geração de relatórios referentes aos funcionários da loja, possibilitando ao usuário o filtro de funcionários para a geração do relatório, podendo escolher entre funcionários que trabalham na loja, e os que já deixaram de trabalhar.

Visando uma futura distribuição do software para que seja implantado em outras lojas, foi vista que na gerencia de uma loja do setor de móveis planejados, é necessário que todo processo realizado pelos funcionários sempre seja documentado de qualquer forma. Portando, com um grande volume de dados à ser armazenado, viu-se a necessidade de que o sistema possua restrições de acesso aos funcionários que serão cadastrados no sistema. Com as restrições de acesso, algumas opções dependendo do funcionário que irá utilizar o sistema, não estará disponível, fazendo com que o software tenha mais foco em seu objetivo, portanto durante o cadastro de cada funcionário, o usuário poderá escolher

se o novo funcionário terá acesso ao sistema. Para este controle, foi escolhido um padrão de restrições, que são: Usuário Master, Usuário Administrador e Usuário Padrão.

Para que o sistema seja mais maleável para o usuário que deseja implantar o sistema, foi incluído o cadastro de cargos para os funcionários. Assim, para cada funcionário, é possível informar qual seria o seu cargo na loja. Esta opção, pode ser cadastrada, trazendo também uma maior flexibilidade na distribuição de cargos, possibilitando a inserção da descrição dos cargos que cada funcionário terá.

Dentre tantas funções implementadas no software, foi vista a necessidade da geração de relatórios, possibilitando ao usuário fazer um filtro dos dados tornando a locomoção dos dados mais fáceis, deixando o software mais útil. Estes relatórios, podem ser gerados a qualquer momento, sendo possível criar filtros, e escolher entre diversos formatos, como por exemplo DOCX, que é uma extensão de arquivo referente ao *software* da *Microsoft*, ODT, que é uma extensão do software *BrOffice*, relatórios em HTML, que são abertas diretamente no navegador, dentre outros formatos que deixam o software bastante robusto e prático para a utilização.

Outro aspecto importante para o software, é a usabilidade. A aparência do software, incluindo cores, fontes, tamanhos, ícones, posições e possibilidade de personalização, é de suma importância para o usuário que utilizará o sistema diariamente, portanto, foi incluído no sistema uma aparência agradável, auxiliando o usuário com ícones referentes as suas respectivas funções e também a possibilidade de personalização da área de trabalho, permitindo ao usuário trocar o papel de parede do software.

Após a aparência do software as funções necessárias do sistema estarem definidas, que são apresentadas no Apêndice (1,2,3), foi iniciada a modelagem do sistema usando os diagramas UML, onde foi definido e visualizado as funções do software antes de começar a escrever as linhas de código.

Depois da modelagem dos diagramas, foi feito um esboço do Banco de Dados que será criado pelo *Hibernate*, nele foram incluídos as tabelas necessárias para o armazenamento dos dados, baseando-se nos diagramas e nos requisitos necessários. Depois de muito claro as funções do sistema, de acordo com os diagramas e o esboço do Banco de Dados, foi iniciado o desenvolvimento do software.

As pastas do sistema, foram separadas de acordo com suas específicas funções. Dentro do pacote *Bean*, foram inseridas as classes Java referentes ao banco de dados. Cada classe é uma tabela do banco e cada coluna é um atributo na classe. Cada classe possui suas respectivas *annotations,* que para o Java, são simplesmente anotações, mas para o *Hibernate,* são configurações passadas para a criação do Banco de Dados.

O arquivo *hibernate.cfg.xml,* possui configurações do banco, onde são inseridos as informações para que seja feita a conexão dentre outras configurações necessárias, como por exemplo o mapeamento das classes referente as tabelas do banco. Dentro do pacote *dao,* encontram-se as classes com os métodos para a persistência com o banco de dados. Para isso, existe uma classe chamada *AbstractDAO.java*, nesta classe possuem métodos responsáveis por persistir os dados em banco, sendo o Java uma linguagem Orientado a Objetos, foi possível uma maior reutilização de código simplesmente herdando métodos e atributos da classe *AbstractDAO.java* para as outas, onde uma classe filha, herda os atributos e métodos de uma classe pai.

Dentro do pacote *gui*, ainda existem os pacotes *cadastros* e *detalhe*. Nestes três pacotes, possuem classes referentes às telas do sistema. Dentro do pacote *gui*, se encontram as classes que herdam a classe *JFrame*, que consiste em telas principais do sistema. Ainda dentro do pacote *gui*, os pacotes *cadastros* e *detalhe*, encontram-se classes que herdam classes *JInternalFrame* do Java, que por sua vez são janelas internas como próprio nome indica. A utilização da classe *JInternalFrame*, foi escolhida pela usabilidade que ela fornece ao usuário, possibilitando instanciar diversas janelas ao mesmo tempo, deixando o software mais agradável e maleável para o usuário.

Para instanciar uma janela *JInternalFrame*, é necessário cria-la dentro de um componente Java chamado *JdesktopPane*, entretanto, sendo a maioria das telas *JinternalFrame*, foi possível perceber a repetição de códigos. Uma resolução encontrada para estre problema, foi a criação da classe *Fachada.java*, nela foi criado também um método abstrato para que a partir dela todas as telas fossem instanciados usando um único método, otimizando o código e tornando-o mais fácil para uma possível manutenção do sistema.

Portanto, ao fim do desenvolvimento, foi obtido um sistema robusto e com muitas funções para que o software seja implantado e que gere lucros para uma loja do segmento de Móveis Planejados, facilitando o trabalho manual dos funcionários, gerenciando todas as informações que são geradas no dia a dia, excluindo quaisquer dificuldades com redundância de nomes de clientes, agilizando a procura por dados, referentes a clientes, endereços, telefones, e-mails, projetos, detalhamento de projetos e ambientes, cargos, funcionários, dentre outros dados que poderão ser cadastrados no software

Resultados

O desenvolvimento do sistema Plano Mobiliário, favoreceu o conhecimento e a prática em todas as etapas do desenvolvimento. Desde o levantamento de requisitos, até a última apresentação do sistema para o lojista que foi entrevistado.

De acordo com o levantamento de requisitos, e os documentos reais da loja, foi vista a grande dificuldade que o lojista tem em gerir todo o processo de venda dos móveis, datas referentes a um estado de um ciclo de vida de um projeto, a procura por telefones que muitas vezes precisam ser rápidas e ágeis, a busca por endereços e datas para efetivar as medidas na casa do cliente dentre outros estados do ciclo de vida do projeto, redundância de nomes, gerenciamentos de pastas e arquivos que são gerados para cada cliente, dentre outras diversas dificuldades que o software desenvolvido tem a capacidade de resolver,

Sendo assim, o software possibilita ao usuário, um controle total sobre o ciclo de vida do projeto, tendo em mãos todas as informações necessárias para a gerencia de diversos projetos ao mesmo tempo, sendo possível gerar relatórios de um projeto com grande facilidade, minimizando a necessidade de busca pelas informações no próprio software, sendo possível imprimir os dados dos projetos simplesmente gerando um relatório com diversas opções de extensões de arquivos como: PDF, RTF, ODT, DOCX, HTM, HTML e XLS, estas opções deixam o software mais maleável para o usuário, facilitando o trabalho manual do dia a dia.

Segundo o lojista entrevistado, o software possui grande eficácia, e trará para a loja maior organização e mais facilidade no trabalho cotidiano dos funcionários. Com a

futura implantação do software, a loja terá mais controle dos dados e possibilitará ao lojista, maior conforto e qualidade em seu atendimento.

Referências:

Arnold, K. (2007) "A linguagem de programação Java" / Ken Arnold, James Goslin, David Holmes; tradução Maria Lúcia Blanck Lisbôa – 4. Ed. – Porto Alegre: Bookman.

Araújo, E. M. T. & Batista, M. L. S. (2007) "Uma visão sobre a Qualidade dos Dados", Disponível em <http://www.devmedia.com.br/uma-visao-sobre-a-qualidade-dos-dados/6973>, acessado em 20 maio 2013.

BNDES (2013), "Panorama do setor moveleiro no brasil, com ênfase na competitividade externa a partir do desenvolvimento da cadeia industrial de produtos sólidos em madeira" http://www.bndes.gov.br/SiteBNDES/export/sites/default/bndes_pt/Galerias/Arquivos/conhecimento/bnset/set801.pdf, Outubro.

Core Java, volume I: fundamentos / Cay S.Horstmann, Gary Crnell; tradução Carlos Schafranski e Edson Furmankiewicz; revisão técnica Nivaldo Foresti – 8. Ed –São Paulo: Pearson Prentice Hall. 2010.

Cybis, Walter, Ergonomia e usabilidade: conhecimentos, métodos e aplicações / Walter Cybis, Adriana Holtz Betiol, Richart Faust. – 2. Ed – São Paulo: Novatec Editora, 2010.

Gabardo, A.(2012), PHP e MVC: com codelgniter / Ademir Cristiano Gabardo. -- São Paulo: Novatec Editora.

Horstmann, C. (2005), Conceitos de computação com o essencial de Java / Cay Horstman. Trad. Werner Loeffler. – 3ed. – Porto Alegre: Bookman, 2005.

iReport (2013), "What is iReport", http://ireport-tutorial.blogspot.com.br/2008/11/what-is-ireport.html, Outubro.

Hibernate (2013), "About hibernate", http://www.hibernate.org/about.html, Outubro.

Melo, Ana Cristiana (2010), "Desenvolvendo aplicações com UML 2.2: do conceitual a implementação / Ana Cristiana Melo." 3 ed. Rio de janeiro: Brasport, (pág. 34)

NetBeans IDE (2013), "Bem-vindo ao Projeto NetBeans" https://netbeans.org/welcome_pt_BR.html, Outubro.

OnJava. (2005) "What Is Hibernate", http://www.onjava.com/pub/a/onjava/2005/09/21/what-is-hibernate.html , Outubro .

Pressman, Roger S. (2011), Engenharia de software: uma abordagem profissional / Roger S. Pressman; tradução Ariovaldo Greiesi, Mario Moro Fecchio; revisão técnica Reginaldo Arakaki, Julio Arakaki, Renato Manzan de Andrade. – 7. Ed. – Porto Alegre: AMGH.

Santos, Rafael (2013), Introdução à programação orientada a objetov usando Java / Rafael Santos. – Rio de Janeiro. Elsivier, – 12° reimpressão.

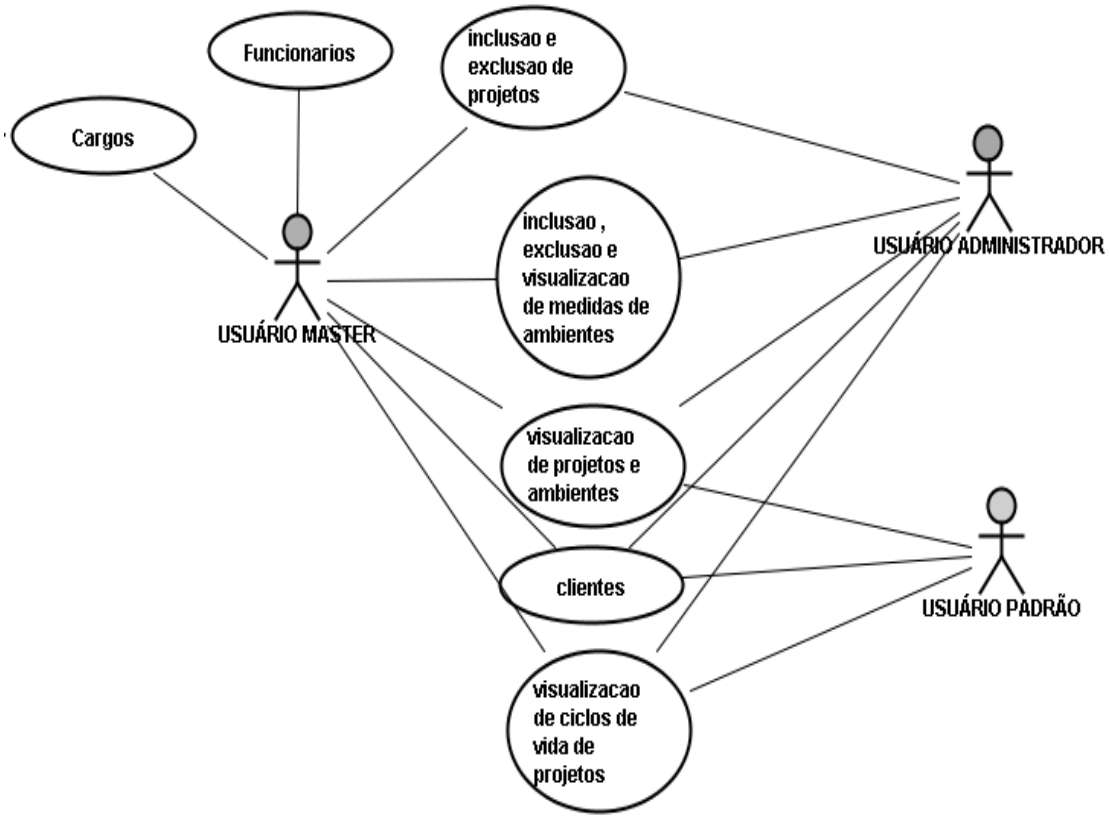
Sommerville, Ian (2011), Engenharia de Software / Ian Sommerville; tradução Ivan Bosnic e Kalinka G. de O. Gonçalvez; revisão técnica Kech Hirama. – 9 ed. – São Paulo:Pearson Prentice Hall.

SEBRAE (2011) http://migre.me/c1u4L, Novembro.

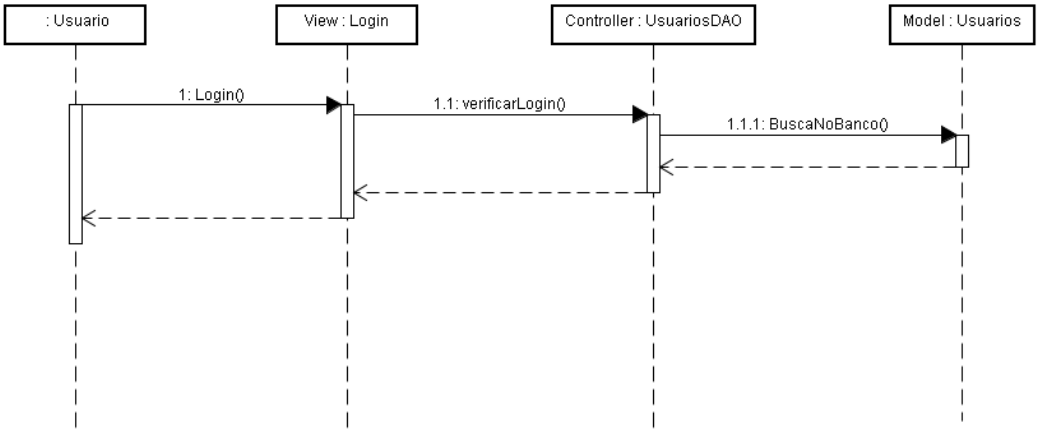
Apêndices

Diagramas UML referentes ao *software* Plano Mobiliário - Sistema de Gestão Empresarial para o Segmento de Móveis Planejados

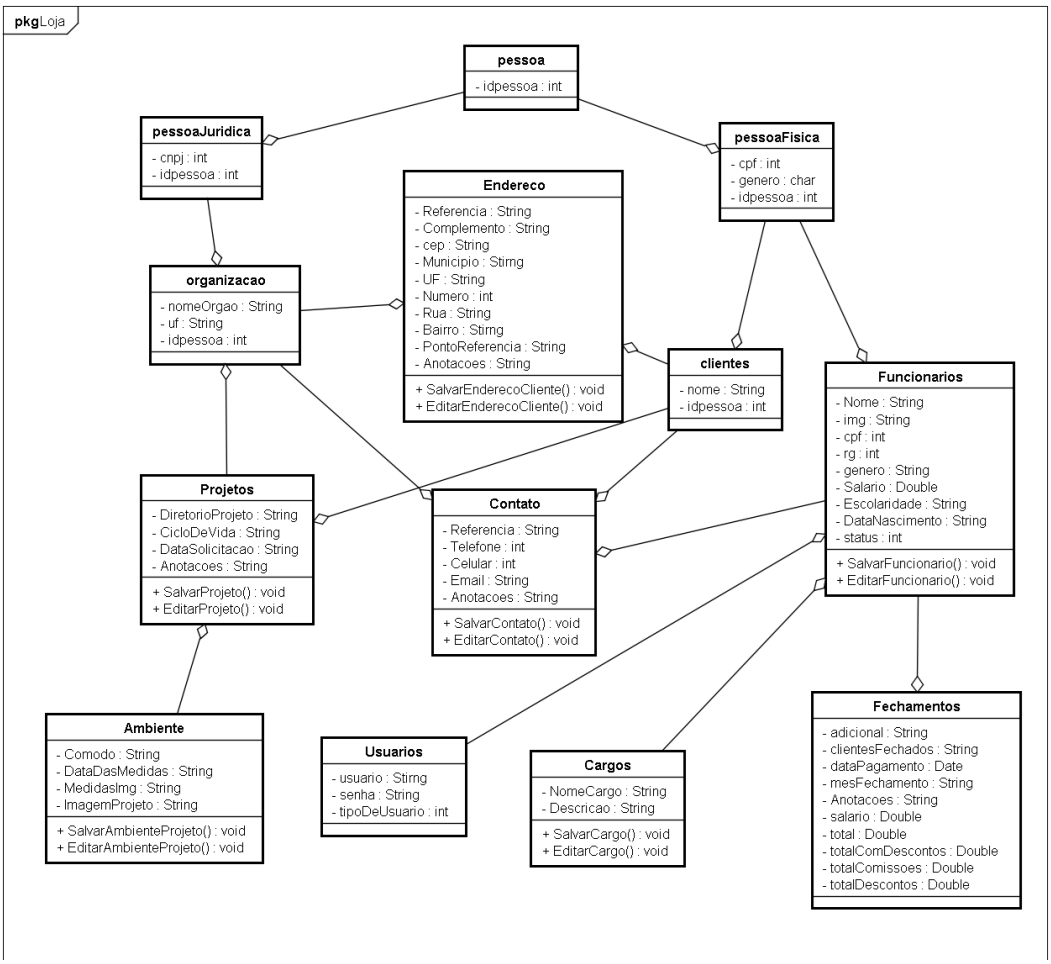
1. Caso de uso:



2. Diagrama de sequência:



3. Diagrama de classe:



4. Banco de dados