Uso da Tecnologia Java no desenvolvimento de um Software para Controle de Produção

Augusto Fernando Sopchuk[1], Willian Ricardo Fialka Agner[2], Leandro Tafuri[3]

[1] Tecnologia em Análise e Desenvolvimento de Sistemas
Faculdade Guairacá - Guarapuava – PR

fernando_bilidi@hotmail.com, will.agner@hotmail.com,
letafuri@hotmail.com

Abstract: *This project was conducted with a view to developing a software to aid in the control of capacity and production time for plywood factory aiming to improve control of its production capacity. Was used the Java Desktop version, and the data were stored in the MySQL database with the help of the Hibernate persistence framework ad SGBD (Systems Managers Database) MySQL Workbench and PhpMyAdmin. The work began with the analysis of the operation of this plant, identified by the user, after interviews, softwares requirements.*

Resumo: Este projeto foi conduzido com vistas a desenvolver um *software* para o auxílio no controle de produção para uma fábrica de compensados visando informatizar seu processo de produção. Foi utilizada a linguagem Java na versão Desktop, sendo que os dados foram armazenados no banco de dados MySQL com o auxílio do framework de persistência Hibernate e os SGBD (Sistemas gerenciadores de banco de dados) *MySQL Workbench* e *PhpMyAdmin*. O trabalho se iniciou com a análise do processo de funcionamento desta fábrica, sendo identificadas junto ao usuário, após entrevistas, os requisitos do *software.*

Introdução

Segundo a ABIMCI (Associação Brasileira de Indústria de Madeira Processada Mecanicamente), em 2009, o Brasil chegou a exportar 662,269 m³ sendo que o principal destino foi os Estados Unidos com 60,03% do total daquele ano. Porém, segundo dados publicados recentemente, na edição número 134 de março de 2013 da Revista da Madeira, as exportações brasileiras de produtos de base florestal continuam a ter dificuldades para atingir níveis positivos devido à crise mundial, no entanto, as empresas contornam esta dificuldade se aproveitando do momento favorável do mercado interno e o maior estado exportador dos itens de madeira continua sendo o Paraná com cerca de US$ 724 milhões. O Estado do Paraná lidera o ranking de área plantada de Pinus com 39,7% da área total, seguido por Santa Catarina, que possui 34,5% (ABRAF, 2013).

Nesse cenário, esta pesquisa se insere e, portanto, buscou-se junto a uma empresa deste ramo, o levantamento, por meio de entrevistas com seus administradores, dos requisitos do *software* a ser desenvolvido.

Para o desenvolvimento deste sistema foi empregada a linguagem de programação Java, a qual é orientada a objetos integrada ao ambiente de

desenvolvimento *Netbeans*. Os diagramas de *software* foram criados no programa *Astah*, o banco de dados escolhido foi o MySQL sendo que para sua persistência foi utilizado o *framework* Hibernate e SGBD utilizados foram o MySQL *Workench* e o *PhpMyAdmin*, o primeiro para a criação da modelagem entidade-relacionamento (MER) e o segundo para a manipulação da modelagem física do banco, já para a construção de relatórios, foi utilizado o *framework JasperReports* em seu ambiente de desenvolvimento *Ireport*.

Ao final deste trabalho, com o sistema elaborado, o usuário poderá realizar o cadastramento de produtos, pedidos, fazer uma programação de produção do pedido e emitir ordens de produção, além de efetuar relatórios de pedidos e ordens de produção.

Fundamentação Teórica

A *Sun Microsystem*, segundo Deitel e Deitel (2010), anunciou o Java formalmente em um encontro do setor em maio de 1995. As atenções foram voltadas para o Java por causa do enorme interesse na *Web*. O Java agora é usado para desenvolver aplicativos coorporativos de grande porte, melhorar a funcionalidade de servidores da *Web*, disponibilizar aplicativos para dispositivos de consumo popular e para muitos outros objetivos.

Melo (2010) nos diz que o Java foi desenvolvido com um subconjunto da linguagem C++, pela *Sun Microsystem*. Sabendo-se que era em larga escala em aplicações para Internet, Intranets e Extranets, tornou-se rapidamente popular.

Segundo Horstmann e Cornell (2010), o Java nunca foi apenas uma linguagem qualquer. Atualmente existem muitas linguagens de programação disponíveis e, poucas talvez nem mesmo se aproximam do poder do Java. Ela é uma plataforma integral, com uma biblioteca imensa com uma grande quantidade de códigos disponíveis para a reutilização e um ambiente de execução que fornece serviços com segurança, portabilidade, para vários tipos de sistema operacionais.

Segundo Gradvohl (2008, p. 4):

> A arquitetura ou plataforma Java tem, basicamente, dois componentes: a máquina virtual Java (JVM) e a interface de programação de aplicações ("*Application Programming Interface*" - API). A JVM, como o próprio nome diz, emula um ambiente computacional. A máquina virtual é a principal responsável pela portabilidade que a plataforma Java provê, pois uma vez instalada em um ambiente computacional real, qualquer programa Java pode ser executado sobre a JVM.

Conforme Horstmann (2005), a linguagem Java apresentou um crescimento bastante significativo, pois os programadores a consideram uma linguagem de fácil acesso, ou seja, uma linguagem mais simples, sendo considerada a irmã mais próxima do C++.

> A estrutura de programas e classes em Java segue a organização de linguagens tradicionais como C e C++, mas sem elementos que tornam programas e programação mais complexos. Após o aprendizado dos conceitos básicos de programação orientada a objetos, o estudante da linguagem pode começar a criar aplicativos úteis e complexos. A

> simplicidade se reflete também na maneira com que arquivos contendo programas em Java são compilados e executados: se as recomendações básicas forem seguidas, o compilador se encarregará de compilar todas as classes necessárias em uma aplicação automaticamente, sem necessidade de arquivos adicionais de configuração e inclusão de bibliotecas. (SANTOS, 2003, p. prefácio vi)

Moreira Neto (2009) nos diz que o modelo orientado a objetos usa como suporte a execução de métodos (pequenas funções que agem normalmente sobre os dados de um objeto), dando importância à maneira como o usuário enxerga o sistema e suas funcionalidades.

A programação orientada a objetos (POO) é o paradigma de programação que atualmente domina o mundo da programação, tendo substituído, conforme nos evidenciam Horstmann e Cornell (2010), as técnicas da programação estruturada que foram criadas nos anos 1970. A linguagem Java é totalmente orientada a objetos, mas o desenvolvedor para utilizá-la precisa estar totalmente habituado a POO.

> Por causa das diferenças estruturais e conceituais, o modelo orientado a objetos é mais natural (mais próximo de nossa compreensão da realidade), e mais econômico no sentido de que necessita de menos passos para atingir um mesmo resultado. (MOREIRA NETO, 2009, p.8).

Melo (2010) nos diz que em meados dos anos 1970, os métodos orientados a objetos começaram a aparecer reforçando ainda mais a mudança de paradigma. A programação orientada a objetos já estava amadurecendo, bastava então criar para ela um caminho a se seguir e então deu início à análise orientada a objetos. O grande ganho com esse novo paradigma é o fato de que se padronizaram os modelos usados para análise, projeto e implementação. Ainda os principais modelos são usados em todas as fases, mudando-se apenas a visão.

Deitel e Deitel (2010) explicam que, em 1980, as organizações começaram a utilizar a Orientação a Objetos para criar aplicativos e desenvolver a OOAD (*Object-Oriented Analysis and Design*) padrão, onde muitos metodologistas as produziram com processos separados para essas necessidades, onde cada processo possuía sua própria notação para transportar os resultados de análises e design. Assim, podendo modelar o software utilizando semelhanças descrevendo objetos da realidade aperfeiçoando o desenvolvimento do protótipo.

Ainda, segundo Deitel e Deitel (2010), a programação orientada a objetos introduz a característica de herança que é como se reutiliza um software de uma nova classe para aprimorá-lo com uma classe existente e com isso economiza-se tempo para o desenvolvimento de novas classes e aumenta a probabilidade de um sistemas ser implementado e mantido efetivamente.

NetBeans IDE é um ambiente de desenvolvimento - uma ferramenta para programadores escrever, compilar, depurar e implantar programas. É escrito em Java - mas pode suportar qualquer linguagem de programação. Existe também um enorme número de módulos para aprimorar o *NetBeans* IDE. O *NetBeans* IDE é um produto gratuito sem restrições de como ser utilizado (NETBEANS, 2013).

Para Paula Filho (2009), na engenharia de software, os processos podem ser definidos para tarefas como desenvolvimento, manutenção, aquisição e contratação de software. Em um processo de desenvolvimento de sistemas, o ponto inicial para a arquitetura deste processo é a escolha de modelo de ciclo de vida.

Paula Filho (2009) ainda evidencia que no modelo de ciclo de vida em Cascata, os principais subprocessos são executados em rígida sequência, por isso é possível demarcá-los com pontos de controle muito bem definidos. Esses pontos de controle facilitam bastante a gerência dos projetos, o que faz com que esse processo seja, em princípio, confiável e utilizável em projetos de qualquer tamanho. "O modelo cascata pode ser muito útil para ajudar desenvolvedores a descrever o que eles precisam fazer. Sua simplicidade o torna fácil de explicar aos clientes não familiarizados como desenvolvimento de software" (PFLEEGER, 2004, p. 39)

Segundo Eckstein (2007), o *Model*, *View*, *Controller* (MVC), foi utilizado pela primeira vez por Trygve Reenskaug, um desenvolvedor do Centro de Pesquisas da Xerox em Palo Alto (EUA), em 1979, e tem como objetivo ajudar a separar o acesso a dados e regras de negócio da maneira em que é exibida para o usuário. Mais precisamente, o MVC pode ser dividido em três elementos:

- ***Model*** (modelo) - O modelo representa os dados e as regras que governam o acesso e atualização de dados. Em uma empresa, um modelo muitas vezes serve como uma aproximação de um processo do mundo real software.
- ***View* (visualização)** - A visão retorna o conteúdo de um modelo. Ele especifica exatamente como os dados do modelo devem ser apresentados. Se há alterações de dados no modelo, a visão deve atualizar sua apresentação, se necessário.
- ***Controller* (controlador)** - O controlador traduz as interações do usuário com a visão em ações que o modelo irá executar.

A figura abaixo demonstra que a entrada do usuário, a modelagem do mundo externo e o feedback visual para o usuário são separados e gerenciados pelos objetos Modelo (*Model*), Visão (*View*) e Controlador (*Controller*).

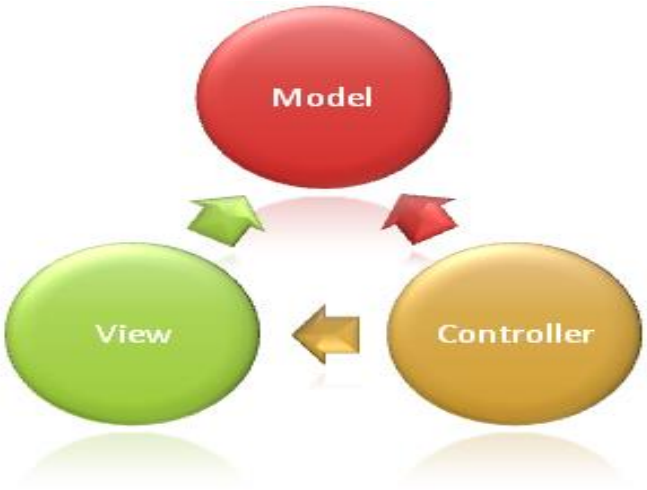


Figura 1: Modelo MVC (Model, View, Controller)

King (2011, *et al*) nos diz que o trabalho com software objeto relacional e banco de dados relacional, pode ser desagradável e cansativo nos dias de hoje em uma empresa e que o Hibernate é um Objeto/Relacional de Mapeamento de ferramentas nos meios Java.

O Hibernate não cuida apenas do mapeamento das classes de Java até as tabelas de banco de dados, mas também assegura, conforme salienta King (2011, *et al*) a consulta de dados e simplicidade para recuperação que pode influenciar e muito na redução do tempo de desenvolvimento.

> A meta de Hibernate é aliviar o desenvolvedor em 95% de dados comuns de persistência relacionados às tarefas de programação. O Hibernate talvez não seja a melhor solução para as aplicações centradas em dados, das quais apenas usam procedimentos armazenados para a implementação das lógicas comerciais no banco de dados. Isto é mais utilizado orientando o objeto aos modelos de domínio e lógicas comerciais na camada intermediária baseada em Java. (KING *et al*., 2011, p.11).

Para Elmasri e Navathe (2011), os bancos e suas tecnologias, têm um peso importante sobre o aumento do uso dos computadores. E é certo afirmar que os bancos de dados realizam um papel essencial em quase todas as áreas em que os computadores são utilizados, incluindo negócios, comércio eletrônico, engenharia, medicina, genética, direito, educação, biblioteconomia.

O MySQL atualmente é o mais popular banco de dados livre do mundo e é desenvolvido, distribuído e suportado pela *Oracle Corporation*. O MySQL oferece um rápido e robusto servidor de banco de dados SQL (*Structured Query Language*) multi-threading e multi-usuário. (MySQL Team, 2013)

Para King (2011, *et al*), no Hibernate também é possível realizar consultas utilizando o dialeto SQL nativo de seu banco de dados, pois ele permite que que o programador especifique o SQL escrito à mão, incluindo procedimentos armazenados, para todas as operações como criar, atualizar, deletar e carregar. Ela também oferece uma caminho de migração limpo de uma aplicação em SQL/JDBC direta até o Hibernate.

Rodrigues Filho (2007) nos diz que o JDBC é a sigla que representa: *Java Database Connectivity* e é a biblioteca de classes de Java que dá acesso ao banco de dados. Ela tem um conjunto de interfaces que criam um ambiente comum em que as aplicações e os mecanismos de banco de dados se comunicam.

Ainda Elmasri e Navathe (2011) nos dizem que o SQL, é uma linguagem de banco de dados imensa: tem comandos para definição de banco de dados, consultas e atualizações. “SQL é uma linguagem de definição e de manipulação de dados relacionais, desenvolvida nos laboratórios da IBM (*International Business Machines*) nos anos 70 e hoje padronizada pelos comitês ISO/Ansi”. (GUIMARÃES, 2003, p. 99)

Oliveira (2002) nos evidencia que a linguagem SQL não é uma linguagem de programação independente, ela poderia ser chamada de sublinguagem, pois quando se desenvolvem aplicações para banco de dados, é preciso utilizar uma linguagem de programação tradicional e utilizar comandos SQL para manipular os dados.

Moreira Neto (2009) afirma que, geralmente, os dados usados e as informações que resultam dos processamentos devem ser salvos em um repositório, que pode ser um arquivo de texto, um arquivo binário, ou, de forma mais tecnológica, um banco de dados relacional. O banco é usado em 95% dos sistemas comerciais por disponibilizar repositórios de dados seguros e robustos.

Segundo Benthin (2010), o MySQL Workbench é uma ferramenta gráfica para modelagem, integrando, criação, designer e administração de banco de dados MySQL. A ferramenta possibilita trabalhar diretamente com objetos *schema*, além de fazer a separação do modelo lógico do catálogo de banco de dados. Combina características profissionais e uma interface clara e simples para oferecer a forma mais eficiente de lidar com as suas bases de dados.

O PhpMyAdmin é uma ferramenta de software livre escrito na linguagem PHP (*PHP: Hypertext Preprocessor*), destinado a lidar com a administração do MySQL pela Internet, suporta uma ampla gama de operações em MySQL. As operações mais usadas (gerenciar bancos de dados, tabelas, colunas, relações, índices, usuários, permissões, etc) podem ser realizadas através da interface do usuário. (PHPMYADMIM, 2013)

Booch (2000, *et al.*) nos diz que a *Unified Modeling Language* (UML) é uma linguagem gráfica padrão muito utilizada no ramo de desenvolvimento de *softwares*, ela é destinada à especificação, construção, visualização e documentação de sistemas, criando assim uma linguagem comum entre desenvolvedores e administradores de *softwares* e WebSites. Essa linguagem de modelagem é a maneira utilizada para expressar quais passos seguir para desenvolver um projeto.

Segundo Bezerra (2006, p.15), o desenvolvimento da UML teve muitos contribuintes, mas os principais foram Grady Booch, James Rumbaugh, Ivar Jacobson. “Finalmente, em 1997, a UML foi aprovada como padrão pelo OMG. Desde então, a UML tem tido grande aceitação pela comunidade de desenvolvedores de sistemas”. A UML é uma linguagem visual para desenhar sistemas orientados a objetos. Quer dizer que a UML é uma linguagem que serve para determinar os elementos visuais que podem ser usados no desenvolvimento do sistema.

Ainda Bezerra (2008) nos diz que a parte visual produzida no percurso de desenvolvimento de um sistema de software orientado a objetos (SSOO), podem ser determinados pelo uso dos diagramas da UML.

Melo (2010) afirma que a UML é uma linguagem para especificação, visualização, construção e documentação de requisitos de sistemas.

Lima (2011, p 30) revela que a:

> UML além de flexível também é extensível e independente de processos ou linguagens de programação, o que garante liberdade para o desenvolvedor adotar qualquer processo, metodologia ou linguagem de programação sem deixar de expressar-se claramente para os seus usuários e outros desenvolvedores, pois utiliza uma notação padrão, comum a todos os ambientes e empresas. Por ser um conjunto das melhores práticas, ela tem suporte de alcance mundial

De acordo com Corrêa (2001, *et al.*), a necessidade de se fazer o planejamento de necessidades futuras de capacidade deve-se a uma essencial característica dos processos de decisão que envolvem obtenção de recursos, ou seja o tempo necessariamente tem de decorrer entre o momento da tomada de decisão e o momento em que a empresa está sentido os efeitos da decisão.

Conforme Corrêa (2001, *et al.*, p 80):

> Independente da lógica que utilize, os sistemas de administração da produção, para cumprirem seu papel de suporte ao atingimento dos objetivos estratégicos da organização, deve ser capazes de apoiar o tomador de decisões logísticas a:
>
> - Planejar as necessidades futuras de capacidade produtiva da organização.
> - Planejar aos matérias comprados.
> - Planejar níveis adequados de estoques de matérias-primas, semi-acabados e produtos finais, nos pontos certos.
> - Programar atividades de produção para garantir que os recursos produtivos envolvidos estejam sendo utilizados, em cada momento, nas coisas certas e prioritárias.
> - Ser capaz de saber e de informar corretamente a respeito da situação corrente dos recursos (pessoas, equipamentos, instalações, materiais) e das ordens (de compra e produção).
> - Ser capaz de prometer os menores prazos possíveis aos clientes e depois fazer cumpri-los.
> - Ser capaz de reagir eficazmente.

Segundo Corrêa (2001, *et al*), as principais ideias para gestão de estoque de certo item são referidos, a quando e quanto ressuprir (via compra, para itens comprados ou produção, para itens fabricados internamente) este item, à medida que ele vai sendo consumido pela demanda.

> Para regular diferentes taxas de suprimento pelo fornecedor e demanda pelo processo de transformação. As taxas diferentes ocorrem, por vários motivos: o fornecedor pode ser pouco confiável e não entregar ou no prazo ou nas quantidades esperadas. (CORRÊA, *et al.*, 2001, p.51).

Desenvolvimento

Este trabalho teve seu início com o levantamento de requisitos que onde após entrevistas foi apontado a necessidade de um software controle de produção. Em seguida foi realizada a análise de requisitos para identificar as funcionalidades e regras de negócio do software.

Após a realização da análise de requisitos, foi dado início aos diagramas da UML, o primeiro a ser feito foi o diagrama de Casos de Uso conforme o anexo 1 (Anexo 1: Diagrama de Casos de Uso), sendo seguido pelo diagrama de Classes conforme verificamos no anexo 2 (Anexo 2: Diagrama de Classes) e por fim o diagrama de Sequência (Anexo 3: Diagrama de Sequência). Tendo como base o diagrama de Classes foi gerada a modelagem relacional do banco de dados (Anexo 4: Modelagem

Relacional do Banco de Dados) com o auxílio da ferramenta *MySQL WorkBench*. Logo após isso, essa modelagem foi exportada para o *PhpMyAdmin* como modelagem física

Com base nessa modelagem, foi dado início ao processo de configuração do Hibernate e o mapeamento das classes que contém os atributos das tabelas foi feito automaticamente por meio de uma classe de entidade de banco de dados que onde é selecionado o banco de dados e o Netbeans cria uma classe mapeando com *annotations* os atributos para cada tabela no banco de dados.

O primeiro pacote a ser criado foi o *Bean* que é onde ficam as classes de mapeamento, as beans, dentro deste pacote estão as classes: ClienteBean, CapProducaoBean, FeriadoBean, ItemPedidoBean, OrdemProducaoBean, PedidoBean, ProgramacaoBean, UnidadeBean e UsuarioBean, cada uma delas contém os seus respectivos atributos que serão mapeados para a geração do banco de dados.

Em seguida foi criado o pacote *Dao* que terá em seu conteúdo as classes: ClienteDao, CapProducaoDao, FeriadoDao, ItemPedidoDao, OrdemProducaoDao, PedidoDao, ProgramacaoDao, UnidadeDao e UsuarioDao, essas classes contém as *criterias* para realizar as buscas no banco de dados com o Hibernate. Em todas as classes *dao* a estrutura dos métodos são as mesmas o que difere uma da outra são os atributos empregados em cada método.

A partir daí foi dado início ao desenvolvimento das telas do sistema que ficarão no pacote *Gui*. A primeira tela a ser desenvolvida foi a tela principal do sistema que é a tela em que ficarão os menus com seus itens onde são as chamadas para as outras telas. Depois deu-se início às telas de cadastro: a primeira tela a ser criada foi a de cadastro de clientes. Vale destacar que quase todas as telas do sistema seguem um padrão de *layout* com cores, tipos de botões e funções o que os diferencia são os dados apresentados. A tela de cadastro de feriados ficou especificada como exemplo abaixo:

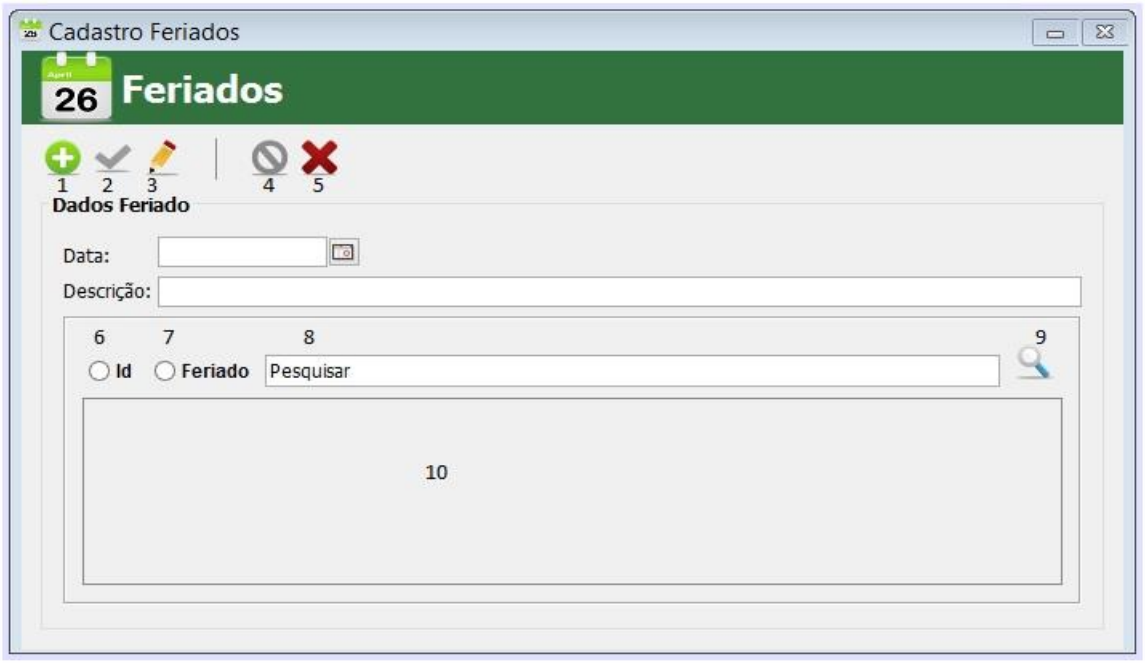


Figura 2: Padrão de Telas do Sistema Cadastro Simples

Outro tipo de tela que existe é a tela de cadastro de pedidos que incorpora também o cadastro de itens de pedido, fato que impede que as pesquisas sejam realizadas na mesma, então há outra tela para a pesquisa de pedidos.

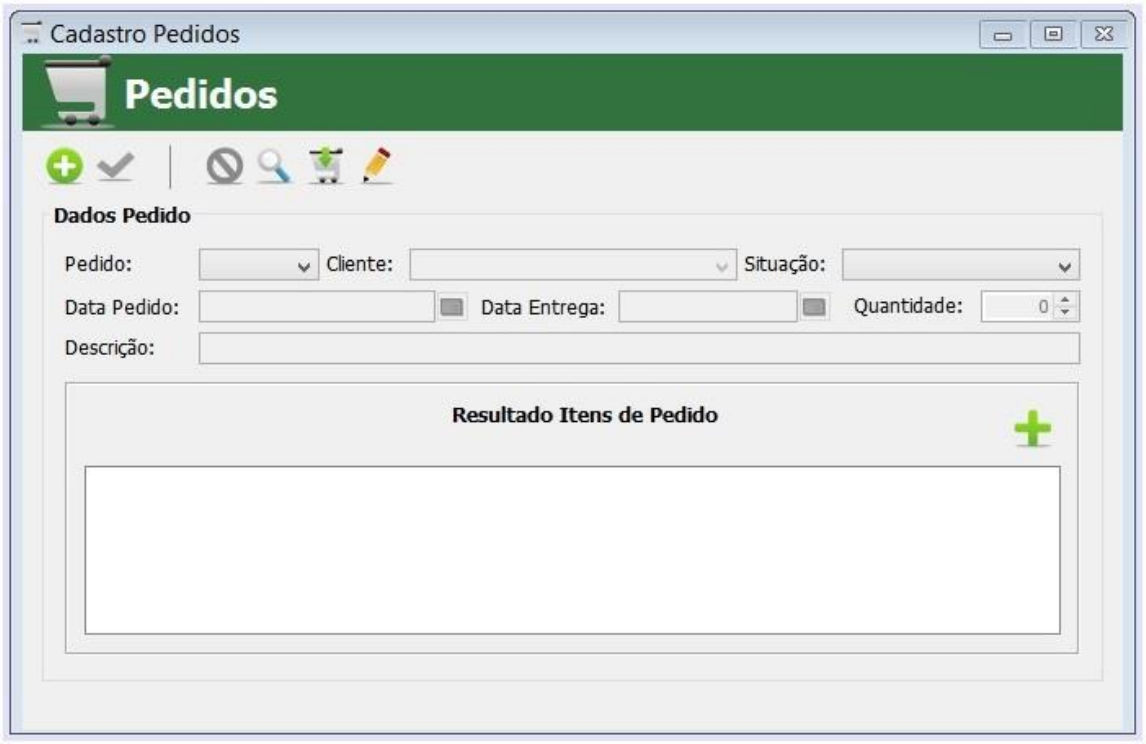


Figura 3: Padrão de Telas do Sistema Cadastro Complexo

Inicialmente as telas vêm com algumas funções desabilitadas e isso poderá ser alterado à medida que o usuário manipula as telas.

As definições das funções padrões seguem abaixo:

1. **Botão Novo**

   Habilita os itens da tela para incluir um registro.
   Habilita os botões de Salvar (2) e Cancelar (4).
   Habilita os campos de inclusão da tela.
   Desabilita os campos de pesquisa e Tabela de Dados (10).

2. **Botão Salvar**

   Grava a inclusão de registros ou a edição dos registros realizados na tela.

3. **Botão Editar**

   Libera a tela para se editar um registro. Antes de se editar um registro, primeiramente uma pesquisa tem de ser executada.
   Habilita os botões de Salvar (2) e Cancelar (4).
   Desabilita os campos de pesquisa e Tabela de Dados (10).
   Habilita os campos de inclusão na tela.

4. **Botão Cancelar**

   Cancela a ação a ser executada no momento.
   Limpa e desabilita os campos de inclusão de registro.
   Habilita os botões Novo (1), Editar (3) e Excluir (5).
   Mantém habilitado os campos de pesquisa e a Tabela de Dados (10) ainda com os itens pesquisados.

5. **Botão Excluir**

   Para se fazer uma exclusão de registro, primeiro deve-se realizar uma pesquisa na Tabela de Dados (10).

   Exclui um item do banco de dados se uma linha da Tabela de Dados (10) estiver selecionada.

6. **Campo de Seleção por Id**

   Se selecionado filtrará e ordenará a tabela por id.

7. **Campo de Seleção por Nome**

   Se selecionado filtrará e ordenará a tabela por nome.

8. **Campo de Pesquisa**

   Campo de texto utilizado para filtro de pesquisa nas buscas com criteria.

9. **Botão Pesquisar**

   Botão utilizado para realizar as pesquisas, seja se o Campo de Pesquisa (8) estiver preenchido ou o Campo de Seleção por Id (6) ou Campo de Seleção por Nome (7) estiverem selecionados ou apenas por um clique no botão ele irá carregar a Tabela de Dados (10) com os registros do banco de dados.

10. **Tabela de Dados**

    Tabela utilizada para a renderização de registros gravados no banco de dados.

Outros tipos de telas e cadastros presentes no sistema são:

Cadastro de Usuário

Contém os dados do usuário como nome, login e senha. Os campos login e senha são necessários para o acesso do usuário ao sistema. Este cadastro é importante para restringir o acesso de pessoas indevidas às informações do sistema.

Cadastro de Feriados

Serão cadastrados por meio desta tela os dados de feriado com data e descrição. Esses dados serão utilizados para a verificação da data de entrega dos pedidos.

Cadastro de Clientes

Nesta tela serão inclusos os dados de cliente como nome, e-mail, telefone, endereço, data de nascimento, tipo de pessoa e cpf/cnpj. Os clientes serão cadastrados apenas como formalidade para ao cadastrar um pedido ter o identificador de para quem esse pedido será entregue.

Cadastro de Produtos

Possui os campos para a inclusão de um novo produto ao sistema, estes campos são unidade, e descrição.

Cadastro de Itens de Pedido

Aqui serão cadastrados os itens do pedido. Essa tela será acessada pela tela de cadastro de pedidos, pois serão os itens serão cadastrados junto com o pedido, essa tela irá conter a medida e o produto a qual está associado.

Cadastro de Pedidos

Nesta tela será realizado um cadastro mais complexo, pois será salvo na mesma a tela os dados de pedido e os itens associados a ele. Os campos presentes aqui serão data do pedido, data para entrega, situação, descrição, quantidade do pedido e os clientes associados ao pedido e uma tabela contendo os itens de pedido além de um combo que contém id do pedido que ao ser selecionado um item deste combo carregue todos os campos da tela além da tabela de item de pedido com os atributos associados a este pedido.

Pesquisar Pedidos

Esta tela contém uma tabela que será carregada com os registros de pedido existentes no banco de dados, além de um combo de cliente que servirá como filtro para a pesquisa dos pedidos.

A tela Pesquisar Pedidos pode ser acessada pela tela de Cadastro de Pedidos ou pelo menu da tela principal do sistema.

Movimentar Pedidos

Tela que é utilizada para aprovação dos pedidos. Contém duas tabela a primeira que terá pedidos com a situação Aguardando Aprovação, ao selecionar um pedido na tabela e clicar em aprovar, será alterada a situação para Aprovado e a linha em questão será enviada para a outra tabela que possui apenas pedidos aprovados.

Cadastro Capacidade de Produção

Contém os campos capacidade de produção por dia, horas trabalhadas, total de produção em horas além de um botão que faz a divisão da capacidade de produção por horas trabalhadas que gera o resultado do total de produção em horas.

Cadastro de Programação

Esta tela serão cadastradas as programações dos pedido. O pedido para ter uma programação deve ter a situação Aprovado. Os campos desta tela são um combo com os pedido aprovados, um campo para situação que o usuário não poderá alterar pois será escolhida a situação “Aguardando Programação” automaticamente ao salvar além dos campos data inicial prevista e data final prevista. Após se cadastrar uma programação o usuário deverá acessar a tela Programação de Produção.

Programação de Produção

Aqui nesta tela serão realizadas os cálculos de quantidade total do pedido, total de tempo de produção e a data prevista para a entrega do pedido. Os campos existentes serão um combo carregado apenas com as programações com situação Aguardando

Programação além dos campos produto, quantidade do pedido, medida total do item de pedido, data de entrega do pedido, data inicial prevista da programação, id do pedido e id do item do pedido. Primeiro será calculado o total do pedido com base neste resultado quantos levará para ser produzido e logo após somado este valor a data de entrega do pedido.

Ordem de Produção

Contém os dados da ordem de produção a ser cadastrada, possui um combo com as programações com situação Programação Finalizada, um campo para situação que o usuário não terá possibilidade de escolha pois ao salvar uma ordem esta situação será salva como Aguardando Produção além de uma data inicial e uma data final.

Emitir Ordem

Esta tela contém uma tabela que recebe as ordens de produção com situação aguardando produção, a partir desta tabela o usuário poderá escolher as ordens a serem emitidas para produção.

Tela de Login

Esta tela foi criada para se ter um controle maior sobre quem terá acesso ao sistema. Primeiramente só terá acesso ao sistema um usuário previamente cadastrado, esses cadastros poderão ser feitos por meio da tela Cadastro de Usuários.

A metodologia de desenvolvimento utilizada neste trabalho foi a de cascata, pois ela se adéqua perfeitamente as etapas que foram realizadas durante o processo de elaboração do *software*, que foram o levantamento de requisitos das necessidades por parte do cliente, avançando pelas fases de planejamento, modelagem, construção, emprego e culminando no suporte contínuo do software já concluído.

Uma das dificuldades apresentadas durante o processo de desenvolvimento deste trabalho foi justamente o fato de o *software* ter de ser eficaz em fornecer suporte nas tomadas de decisões na produção.

Resultados

O objetivo proposto para este trabalho foi o desenvolver um sistema *desktop* denominado de Controle Capacidade de Produção (CCP), para uma fábrica de compensados situado na cidade de Virmond-PR, o sistema foi desenvolvido de acordo com o levantamento de requisitos. O processo que o *software* se propôs a informatizar foi o controle de produção que é a maior dificuldade das empresas do ramo.

Foi utilizada a linguagem Java por se tratar de uma linguagem livre, que possui dinamismo para se trabalhar em qualquer plataforma operacional e possui vários tipos de matérias para acesso com livros, apostilas e fóruns na internet. A persistência no banco de dados foi realizada com ajuda do banco de dados livre MySQL que é um banco de dados que apresenta grande facilidade em sua administração. O Hibernate agilizou e muito o processo da persistência por ser um *framework* de muita praticidade.

O *software* após sua implantação foi capaz de apresentar os resultados esperados pelo cliente e atendeu a todas as necessidades apontadas no levantamento de requisitos, podendo ser capaz de prever o tempo necessário para se produzir um pedido.

Considerações Finais

Após o desenvolvimento deste *software*, ele foi apresentado ao cliente, ele se mostrou satisfeito com a interface e com os cadastros abordados no sistema. O cliente também afirmou que o sistema atendem as necessidades expostas na análise de requisitos, o que pode ser confirmado após implantação do mesmo na empresa.

Durante o processo de desenvolvimento deste trabalho foi possível adquirir conhecimento de como funciona um controle de produção, também se familiarizar com o *framework* Hibernate, além de aprimorar os conhecimentos em Java.

Futuramente será desenvolvido o controle de saída dos produtos com realização de controle de estoque, pois para realizar este desenvolvimento deve se levar em consideração que seria necessário abranger mais produtos e outras regras de negócio.

Referências

Abimci**.** **"**Estudo Setorial 2009 base 2008" Disponível em: <http://www.abimci.com.br/abimcidocs/2009.pdf>. Acesso em 28 out. 2013.

Abraf. "Anuário estatístico da ABRAF 2012 – ano base 2011." Brasília: 2012. 150 p. Disponível em: < http://www.abraflor.org.br/estatisticas/ABRAF12/ABRAF12-BR.pdf> Acesso em: 04 nov. 2013.

Benthin, F, (2010) "Planejamento de bancos de dados com MySQL Worbench." Linux Magazine #67. Disponível em: < http://www.linuxnewmedia.com.br/images/uploads/pdf_aberto/LM_67_50_53_04_ana_mysql.pdf> Acesso em: 04 nov. 2013

Bezerra, E. (2006) "Princípios de análise e projeto de sistemas com UML". 2.ed. Rio de Janeiro: Elsevier.

Booch, G., Rumbaugh, J. and Jacobson, I., (2000) "UML, guia do usuário". Trad. Fábio Freitas da Silva. Rio de Janeiro: Campus,

Corrêa, H. L., Gianesi, I. G, N. and Caon, M. (2001) "Planejamento, Programação e Controle da Produção". 4. ed. São Paulo: Editora Atlas S.A,

Deitel, H. M. and Deitel, P. J. (2010) "Java Como programar". 8.ed. São Paulo: Person Prentice Hall.

Eckstein, R. (2007) "Java SE Application Design With MVC"**.** Disponível em: < http://www.oracle.com/technetwork/articles/javase/index-142890.html >. Acesso em: 29 out. 2013.

Elmasri, R., and Navathe, S. B. (2011) "Sistema de Banco de Dados". 6. ed. Trad. Daniel Vieira. São Paulo: Pearson Addison Wesley.

King, G., Bauer, C., Anderser, M. R., Bernard, E. and Ebersole, S. "Documentação de Referência Hibernate". Disponível em: <http://docs.jboss.org/hibernate/orm/3.5/reference/pt-BR/pdf/hibernate_reference.pdf>. Acesso em 18 out. 2013.

Guimarães, C. C. (2003) "Fundamentos de banco de dados: modelagem, projeto e linguagem SQL." Campinas: Editora da Unicamp.

Gradvoh, A. L. S. (2008) "Introdução à Linguagem de Programação Java." Centro Nacional de Processamento de Alto Desempenho - São Paulo. Copyright. Disponível em: < http://www.youblisher.com/p/416217-Linguagem-Java/ >. Acesso em: 26 out. 2013.

Horstmann, C. S (2005). "Conceitos de computação com o essencial de Java". 3.ed. Trad. Werner Loeffler. Porto Alegre: Bookman.

Horstmann, C. S. and Cornell, G. (2010) "Core Java: Fundamentos." 8.ed. Trad. Carlos Schafranski e Edson Furmankiewicz. São Paulo: Pearson Prentice Hall, v.1.

Lima, A. S. (2011) "UML 2.3: Do requisito à solução." 1. Ed. São Paulo: Érica.

Melo, A. C. (2010) "Desenvolvendo aplicações com UML 2.2: do conceitual à implementação". 3.ed. Rio de Janeiro: Brasport.

Moreira N. O. (2009) “Entendendo e Dominando o Java: Criando classe de objetos a partir de um modelo.” 3.ed. São Paulo: Digerati Books.

MySQL Team. (2013) “Manual de Referência do MySQL 5.5.” Disponível em: < http://dev.mysql.com/doc/refman/5.5/en/index.html >. Acesso em: 15 out. 2013

NetBeans Team (2013) “NetBeans IDE para Java Open Source” Disponível em: <https://netbeans.org/index_pt_PT.html > Acesso em: 04 nov. 2013.

Oliveira, C. H. P. (2002) ”SQL: Curso Prático.” São Paulo: Novatec Editora.

Paula Filho, W. P. (2009) “Engenharia de Software: Fundamentos, métodos e padrões." 3. Ed. Rio de Janeiro: LTC.

Pfleeger, S. L. (2004) “Engenharia de Software: teoria e prática.” 2. ed. Trad. Dino Franklin; revisão técnica Ana Regina Cavalcanti da Rocha. São Paulo: Prentice Hall.

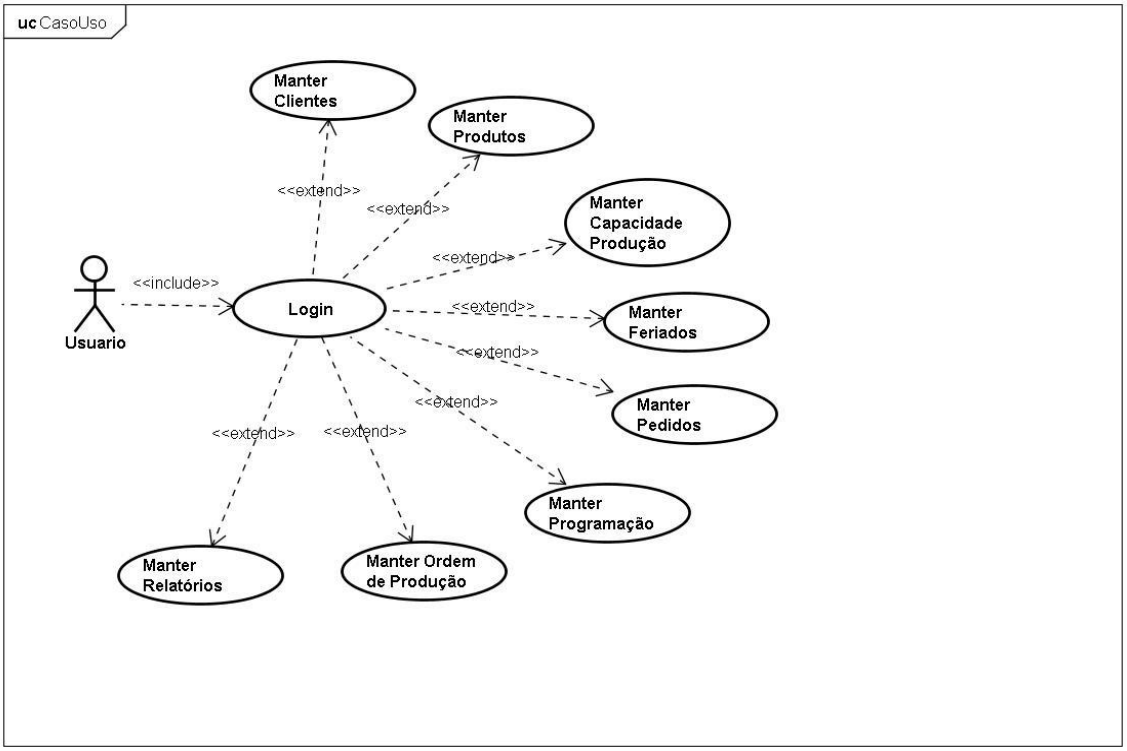
PhpMyAdmin. (2013) “Documentação PhpMyAdmin.” Disponível em: <http://www.phpmyadmin.net/home_page/index.php > Acesso em 04 nov. 2013.

Revista da Madeira. “Exportação.” Disponível em: <http://www.remade.com.br/br/revistadamadeira_materia.php?num=1648&subject=Exporta%E7%E3o&title=Crise%20mundial%20ainda%20limita%20exporta%E7%F5es%20do%20setor>. Acesso em: 28 out. 2013

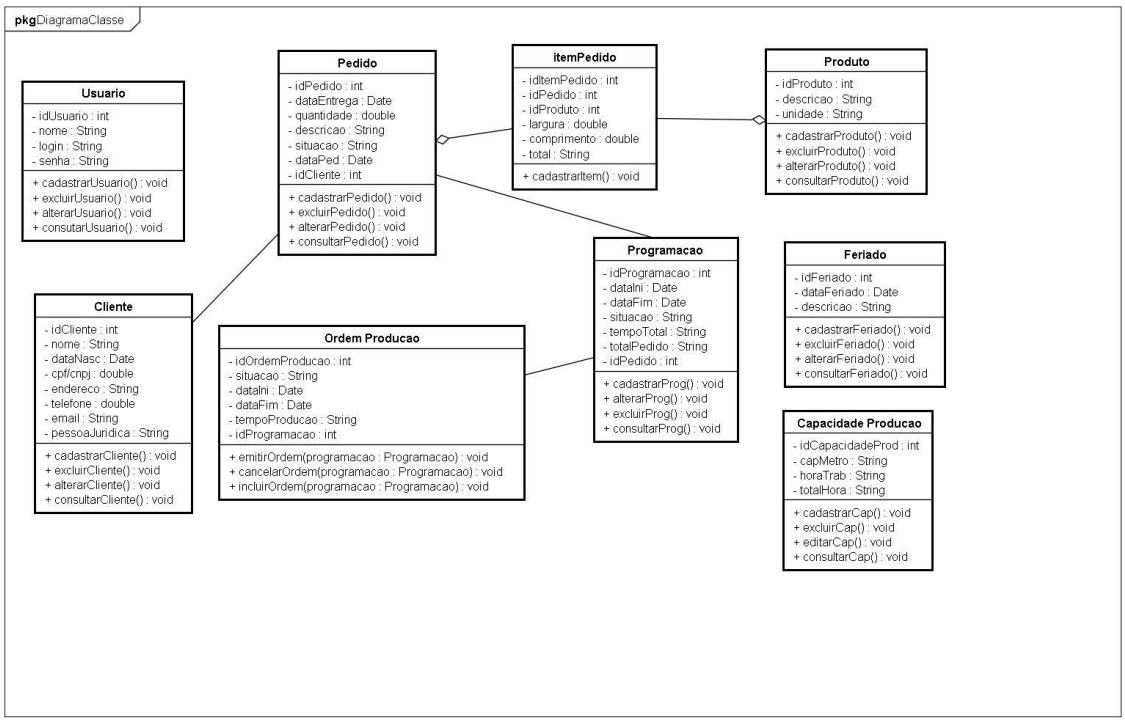
Rodrigues Filho, R. (2007) “Desenvolva aplicativos com Java 2.0” 2.ed. São Paulo: Érica.

Santos, R. (2003) “Introdução à Programação Orientada a Objetos usando JAVA:” São Paulo: Elsevier.

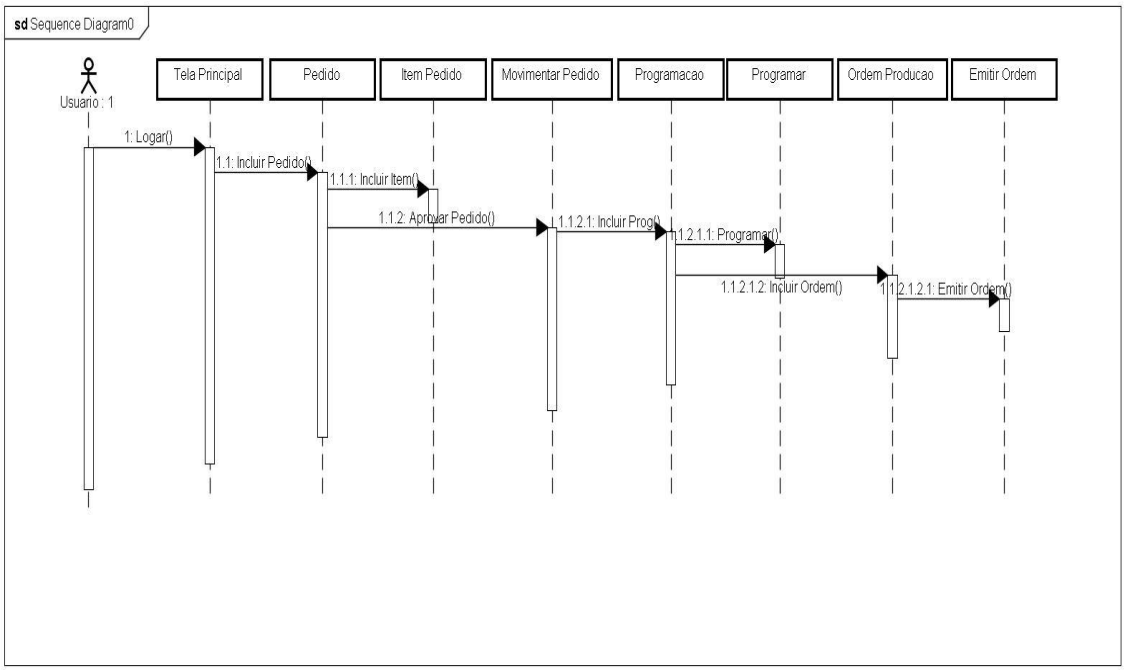
Anexos



Anexo 1: Diagrama de Casos de uso.

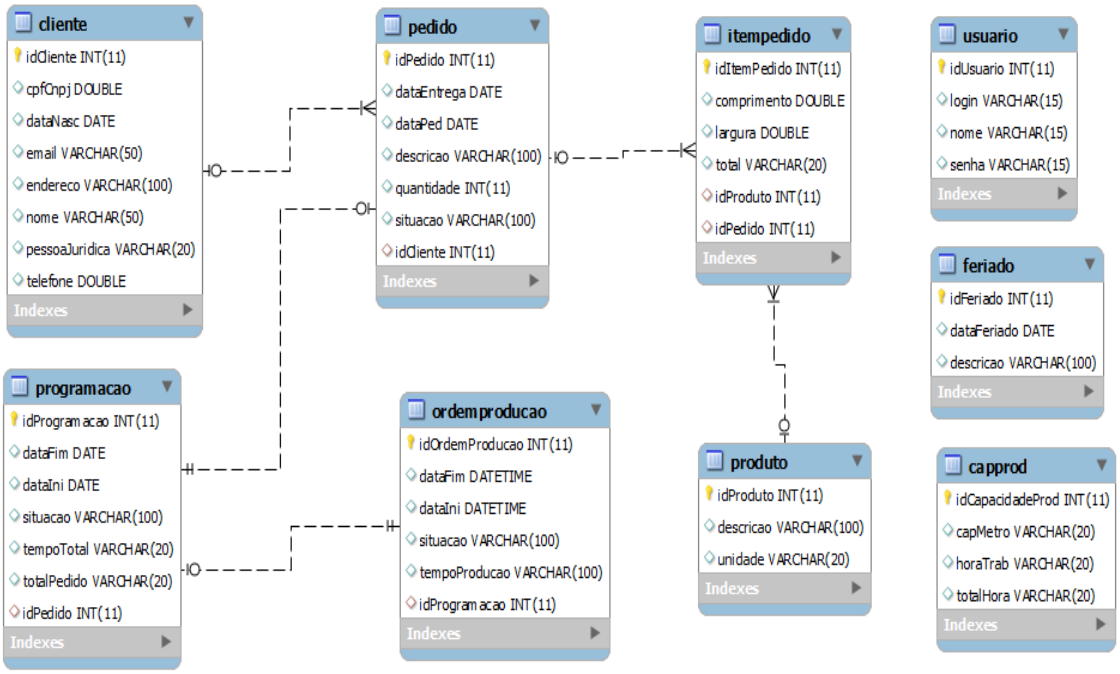


Anexo 2: Diagrama de Classes



Processo de Emissão de Ordem de Produção

Anexo 3: Diagrama de Sequência



Anexo 4: Modelagem Lógica do Banco de Dados.