Utilização da Orientação a Objetos para desenvolvimento de um sistema web para controle de estoque

Gustavo F. Galvão[1], Sergio Ribeiro[2]

[1] Tecnologia em Análise e Desenvolvimento de Sistemas – Faculdade Guairacá CEP 85010-000 – Guarapuava – PR – Brasil

[2] Universidade Estadual de Ponta Grossa, Departamento de Informática, Mestrado em Computação Aplicada em Agricultura

[1]gf_galvao@live.com, [2]sergio.ribeiro@docase.com.br

Abstract. *This article describes the steps followed to develop a system for inventory control via the web that aims to improve methods for the verification of stock. The system was developed using the PHP language, concepts of object orientation, MySQL, CSS, jQuery, Bootstrap, Calango framework data, among other technologies, MVC design pattern and the prototyping life cycle. The requirements for the same were obtained by interview at a company that was also responsible for testing the functionality of the system that were available through prototypes.*

Resumo. *Este artigo descreve os passos seguidos para o desenvolvimento de um sistema para controle de estoque via web que tem como objetivo melhorar os métodos para a verificação de estoque. O sistema foi desenvolvido utilizando a linguagem PHP, conceitos de orientação a objetos, banco de dados MySQL, CSS, Jquery, Bootstrap, Calango framework, entre outras tecnologias, padrão de projetos MVC e o ciclo de vida prototipação. Os requisitos para o mesmo foram obtidos através de entrevista em uma empresa que também foi responsável pelos testes das funcionalidades do sistema que foram disponibilizados através de protótipos.*

1 Introdução

Este trabalho teve como objetivo criar uma aplicação para controle de estoque de acesso remoto. O aplicação é um sistema *web que* permite aos funcionários de uma empresa acesso ao mesmo de qualquer computador com acesso à *internet* para que eles possam verificar os produtos em estoque.

O principal motivo para o desenvolvimento de trabalho é o fato de que a empresa ainda não utiliza ferramentas que disponibilizam funcionalidades especificas para o controle de estoque, utilizando métodos manuais para fazer este controle. Com isso no processo de verificação de mercadoria, por exemplo, de maneira manual um funcionário precisa se deslocar ao local onde o produto desejado se encontra para verificar se há a quantidade necessária do mesmo, o que faz com que o cliente fique esperando, em alguns casos muito tempo, até receber a informação sobre o produto.

Desta forma, planejou-se o desenvolvimento de um sistema com o objetivo principal de resolver os problemas encontrados para a verificação de produtos através de funcionalidades que diminuam esse tempo, e geração de relatórios com objetivo de fornecer um apoio a tomada de decisões da empresa, em relação a compra de produtos.

2 Fundamentação Teórica

Os sistemas de informação são compostos por software, hardware, pessoas, regras e outras tecnologias que trabalham juntas com objetivos comuns, recebendo e processando dados, para produção e distribuição de informações. E fazem parte do dia-a-dia das pessoas, como em portais web, sistemas comerciais em escolas e empresas, com funcionalidades específicas para cada área (SHITSUKA *et al.,* 2005)

Em relação ao desenvolvimento de software, Pressman (2011) chegou à conclusão de que o "software, em todas as suas formas e em todos os seus campos de aplicação, deve passar pelos processos de engenharia". Ainda, segundo ele, a engenharia de software deve ser utilizada para obter software econômicos, confiáveis e que funcione de forma eficiente através de princípios sólidos de engenharia.

A orientação a objetos é um paradigma que representa a filosofia para o desenvolvimento de sistemas, nela é usada uma ótica mais próxima do mundo real ao invés de uma onde o sistema é formado por um conjunto de procedimentos e variáveis que nem sempre estavam agrupadas pelo contexto em que se encontra. Ela trabalha com classes e objetos, onde a classe é uma estrutura responsável por definir o tipo de dado, podendo conter atributos e funções para manipulá-los. É um modelo para a criação de objetos. As classes são orientadas ao assunto, isso porque cada classe é responsável por um assunto diferente(DALL'OGLIO, 2011).

A orientação a objetos tem como base as palavras encapsulamento, herança e polimorfismo sendo que cada uma tem seu papel, mas elas coexistem complementando um ao outro. Com encapsulamento pode-se criar partes ocultas que não serão mostradas no fora do software, o conceito de herança permite que uma classe possa ser construída com base em outra existente herdando assim todas as funcionalidades da mesma e o polimorfismo permite que um único nome de classe/método possa representar códigos diferentes. (DE MELO, 2007).

Padrões de projetos são "descrições de objetos e classes comunicantes que precisam ser personalizadas para resolver um problema geral de projeto num contexto particular". São responsáveis por nomear, abstrair e identificar aspectos-chave de uma estrutura de projeto com o objetivo de fazer que ela se torne útil para o desenvolvimento de um projeto orientado a objetos que seja reutilizável (GAMMA *et al.*, 2006).

Padrão de projeto pode ser caracterizado como "uma regra de três partes que expressa uma relação entre um contexto, um problema e uma solução". O contexto, para o projeto, permite ao leitor compreender o ambiente em que se encontra o problema e qual a solução adequada para o mesmo. Os padrões de projetos identificam classes, instancias que participam, quais seus papéis, colaborações e também a distribuição de responsabilidades. Cada padrão se foca em um problema específico de um projeto, ele descreve quando deve ser utilizado, suas restrições e consequências, seus custos e benefícios. (PRESSMAN, 2011).

O padrão de projeto mais conhecido é o *Model*, *View*, *Controller* (MVC) que é composta por três objetos, sendo eles o objeto de aplicação (Modelo), o objeto de apresentação na tela (Visão) e o Controlador que é o que define como a interface reagirá às entradas do usuário, ou seja, as aplicações que seguir este padrão é, basicamente, dividida em três camadas. Essa separação em camadas traz vantagens para o desenvolvedor, por exemplo ao separar o modelo de dados da visualização ele pode reutilizar o mesmo modelo em visualizações diferentes. Mesmo as três camadas sendo distintas elas interagem uma com a outra, sendo que as camadas *View* e *Controller* dependem da Modelo, mas a esta camada é independe das demais, e *View* e *Controller* dependem uma da outra, pois a controller depende dos dados e ações vindas da mesma e a view aciona a controller e aguarda retorno da mesma (DALL'OGLIO, 2011).

Todo projeto pode ser dividido em fases de desenvolvimento, o que permite a equipe ter um melhor controle dos gastos é o entendimento dessas fases. O conjunto desses fases é conhecido como ciclo de vida, esse ciclo permite identificar similaridades entre projetos, mesmo que estejam em contextos diferentes. O ciclo de vida pode ser dividido em conjuntos de fases que normalmente são fixas, sendo aplicadas para todos os tipos de projetos, essas divisões feitas contêm passos principais do processo de contextualizar, desenhar, desenvolver e colocar em operação as necessidades do projeto. O conhecimento destas fases proporciona vários benefícios para os projetos, sendo que entre elas estão a correta análise do ciclo de vida que determina o que foi e o que não foi feito pelo projeto, avalia como o projeto está progredindo e permite identificar em que ponto o projeto se encontra (VARGAS, 2005).

Ainda, de acordo com Vargas (2005) *apud* Meredith, os ciclos de vida são caracterizados por inícios lentos que é seguido por um processo acelerado até atingir seu pico e começa a decair até chegar ao fim do projeto. O mesmo ocorre com o nível de dificuldade que se inicia-se praticamente nulo e vai crescendo até um nível máximo e vai decaindo até atingir o término do projeto.

Interfaces ergonômicas se tornam difíceis de se desenvolver pelo motivo de constituírem sistemas abertos onde seus usuários são geralmente usuários ativos, cuja maneira de pensar e comportar se modifica tanto como consequência como causa da constante evolução tecnológica. As mesmas funções podem significar coisas diferences para cada usuário dependendo do momento e contexto em que se encontram. Cada usuário tem uma experiência individual e única, pois cada um tem sua bagagem de conhecimento e experiências. A usabilidade não é uma qualidade interna de um sistema, mas depende de um acordo entre as características de sua interface com os usuários, ela caracteriza o uso dos programas e aplicações. Usuários experientes podem ter interações satisfatórias enquanto os novatos deixam muito a desejar na utilização de uma mesma interface. O mesmo é obtido, não importando o tipo de usuário, caso o programa seja executado em um computador rápidos ou lentos, ou dependendo da frequência da execução da tarefa. O acordo entre interface, usuário, tarefa e ambiente é a essência da usabilidade. (CYBIS *et al.,* 2010).

A usabilidade é definido pela norma ISO 9241 como "A capacidade que um sistema interativo oferece a seu usuário, em determinado contexto de operação, para a realização de tarefas de maneira eficaz, eficiente e agradável."

A ergonomia, além de bem-estar e saúde do usuário, proporciona eficiência e eficácia da aplicação, por esse motivo. Cybis *et al,* (2010) diz que ela está na origem da

usabilidade, e para que sistemas e dispositivos proporcionem usabilidade eles devem estar adaptados ao modo que o usuário pensa, age e trabalha. Para que sejam construídas interfaces ergonômicas e que proporcionem usabilidade o usuário e seu trabalho devem ser bem conhecidos pelos desenvolvedores envolvidos no sistema.

Nielsen (1994) propõe as Heurísticas de Usabilidade que é um conjunto de dez qualidades de base para o desenvolvimento para as interfaces. Entre elas estão a consistência e padrões, visibilidade do estado do sistema, entre outras.

A *web* foi inventada em 1992 por Sir Tim Berners-Lee que atualmente é diretor da W3C (*World Wide Web Consortium*), pesquisador sênior do Laboratório da Ciência da Computação e Inteligência Artificial (*CSAIL*) do Instituto de Tecnologia de Massachusetts (*MIT*) e professor de Ciência da Computação na Universidade de Southampton, na Inglaterra. Em 1990, Berners-Lee criou o primeiro navegador para ser executado em computadores da *NeXT*. Ele acreditava que com o uso de links globais seria possível interligar hipertextos em diferentes computadores. Para isso ele desenvolveu o software próprio e protocolo, chamado *HTTP*, para recuperar hipertexto. O texto criado para o *HTTP* foi o *HTML* (*Hypertext Markup Language*) que teve como base para sua criação a especificação *SGML* (*Standard Generalized Markup Language*) que é uma diretriz internacional que contem regras para a criação de linguagens de marcação (SILVA, 2011).

HTML é uma linguagem de marcação cujo objetivo é a descrição dos dados da página e consiste em texto plano e marcações chamadas *tags*, que são comandos da linguagem. Na exibição de uma página web, o navegador recebe o conteúdo e interpreta as *tags* contidas nele para construir a página. A declaração de uma *tag* é feita com os caracteres < e >, onde é inserido entre eles os comandos que serão interpretados pelo navegador. A *tags* podem ser escritas em pares como é o caso da *tag <title></title>* ou escritas sozinhas como é o caso da *tag
* (DE MELO, 2007).

Hipertexto pode ser resumido como "todo conteúdo inserido em documentos para web e que tem como principal característica a possibilidade de se interligar a outros documentos da web". Para que um site seja construído com os padrões da web deve-se saber usar o *HTML* somente para a estruturação das páginas e não utilizar elementos de apresentação, *CSS*, junto com as marcações. Foi criado, em 1991, uma lista de discussão eletrônica chamada *WWW-talk* cujo seu objetivo era trocar ideias e experiências sabre a linguagem desenvolvida por Tim Bernes-Lee (SILVA, 2008).

Em 1993 foi criado a *HTML+*. A *HTML+* começa afirmando que "documentos marcados com *HTML+* são constituídos de títulos, parágrafos, listas, tabelas e figuras".

E continua:

> *"Ao contrário da maioria das tecnologias destinadas à criação de documentos, a HTML+ não se destina a determinar a aparência; assim, nomes e tamanhos de fontes, margens, tabulações, espaçamentos entre os elementos não são funções da linguagem. Fica a cargo dos softwares responsáveis pela renderização dos documentos marcados com HTML+ a maneira como os documentos devam ser apresentados (talvez com base em configurações de preferência do usuário)".*

Desde a criação do HTML seus idealizadores se preocuparam em retirar da mesma qualquer atribuição ou função de apresentação, pois a linguagem de marcação deve ser utilizada exclusivamente para a estruturação de documentos.

A *HTML* tornou-se um caos com a criação dos diversos navegadores, pois cada empresa criava suas próprias formas de marcação e para tentar eliminar os problemas que eram gerados pelas novas formas de marcação Don Connolly e colaboradores levantaram tudo que existia na *HTML* e propuseram em 1994 o *HTML* 2.0, cuja versão final foi publicada em 1995 (SILVA, 2011).

A *W3C* mudou de ideia sobre sua decisão de encerrar o desenvolvimento do *HTML* em favor da *XHTML* e anunciou sua decisão de continuar os estudos para o desenvolvimento do *HTML5* utilizando como base o trabalho da *WHATWG*.

A linguagem de marcação na sua versão atual (4.01) não possui funcionalidade que permitam que seja adicionada interatividade avançada para as páginas, sendo que para este proposito são utilizadas outras linguagens de programação, como o JavaScript (SILVA, 2011).

A definição mais simples e precisa para *CSS* é a da *W3C* que diz "Folha de estilo em cascata é um mecanismo simples para adicionar estilos (por exemplo: fonte, cores, espaçamentos) aos documentos *web*".

A *HTML* não foi desenvolvida para fornecer aos usuários informações sobre a apresentação dos elementos como cores de fontes, tamanhos de textos e todo o aspecto visual de um documento não devem ser função do *HTML*. Todas essas funções citadas cabem à *CSS* (Silva, 2008).

Quando Tim Berners-Lee desenvolveu o navegador *Nexus*, em 1990, ele criou também funcionalidades bastante limitadas que ele utilizava para controlar a apresentação dos documentos. Sendo que no início somente era possível o controle de algumas cores e fontes. As funcionalidades citadas anteriormente hoje em dia são conhecidas como folhas de estilo-padrão do navegador. A primeira proposta para a implementação das *CSS* foram feitas no segundo semestre de 1994. Tim Berners-Lee não publicou a sintaxe utilizada para criar a folha de estilo-padrão de seu navegador por achar que essa era uma questão a ser resolvida pelo navegador. A *W3C* lançou em 1996 as *CSS1* como uma recomendação oficial (SILVA, 2011).

O *CSS* foi criado para complementar o *HTML* e é utilizado para descrever como os documentos devem ser apresentados e podem alterar os mesmos de diferentes maneiras. *Ele* pode estar tanto na composição da página *HTML* quanto em um arquivo individual que deve ser referenciado na composição do arquivo *HTML*. *A* utilização do *CSS* veio para dizer que o código *HTML* deveria conter apenas descrições de dados e não a formatação (DE MELO, 2007).

O *JavaScript* foi criada pela *Netscape* em parceria com a *Sun Microsystems* com o objetivo de oferecer uma maneira de adicionar interatividade as páginas *web*. Sua primeira versão, *JavaScript* 1.0, foi lançada em 1995 e implementada em março de 1996 no navegador *Netscape Navigator* 2.0. Em resposta à *Netscape*, a *Microsoft* criou a linguagem *JScript* que era baseada em *Visual Basic*, sua primeira versão foi lançada para o *Internet Explorer* 3.0. Atualmente seu nome oficial é *JavaScript* é *ECMAScript* e a versão é a *ECMA-262 v5* (SILVA, 2010).

HTML é limitada a fornecer rótulos e campos de um formulário que serão preenchidos pelo usuário, com ele não conseguimos processar os dados nem mesmo enviá-los, sendo que para executar essas necessidades são utilizados outros programas que possam manipular e processar dados, entre as linguagens de programação que executam estas funções destacam-se *PHP, ASP, Java, Ruby, Python e ColdFusion*. Estas linguagens foram desenvolvidas para serem executadas no lado servidor. O *JavaScript* foi desenvolvido para ser executado no lado cliente, isto é, sua interpretação e funcionamento dependem do navegador utilizado que tem um interpretador *JavaScript* hospedado no mesmo (SILVA, 2010).

Com o *JavaScript* pode-se manipular o conteúdo e apresentação de páginas HTML já existentes, manipular o navegador, interagir com formulários, interagir com outras linguagens dinâmicas.

Javascript deve ser utilizada em conjunto com o *HTML* para que ela ajude na interação com os usuários e também realizar diversas tarefas que são executadas no lado cliente, como por exemplo apresentar mensagens sobre verificação dos campos de um formulário (DE MELO, 2007).

Para fazer *scripts* desenvolvidos em *JavaScript* precisamos apenas de um navegador, o que não ocorre em programas escritos na linguagem *PHP* onde eles precisam estar hospedados em servidores remotos configurados para rodar *PHP* ou em servidor local com suporte para o mesmo (SILVA, 2010).

JQuery é uma biblioteca *JavaScript* criada por John Resig em janeiro de 2006 em Nova York. Silva (2010) *apud* John Resig diz que "o foco principal da biblioteca *JQuery* é a simplicidade. Por que submeter os desenvolvedores ao martírio de desenvolver longos e complexos códigos para criar simples efeitos? ".

Para Silva (2010) o *JQuery* é uma maneira simples e fácil de escrever *JavaScript* que pode ser utilizado tanto por programadores experientes quanto por profissionais que pouco sabem sobre programação.

O *JQuery* tem como lema "*Write less, do more*" (Escreva menos, faça mais), um exemplo dessa aplicação é substituir várias linhas de código que seriam utilizadas para aplicar um simples efeito em um componente por apenas algumas com sintaxe *JQuery*.

Desde o surgimento da internet a interação do usuário com servidor via *HTTP* era através de sistemas simples de hipertexto, onde quando clicava em um link era enviado uma requisição para o servidor que após processar essa requisição respondia devolvendo o documento, após utilizar este documento o usuário clica em outro link e espera novamente. Este sistema ainda é utilizado nos dias de hoje (Niederauer, 2007).

Segundo Niederauer (2007) mesmo com a evolução dos navegadores havia muitas restrições, que fizeram com que os desenvolvedores trabalhassem para melhorar o modelo de interação da *web*. Com isso eles criaram novas maneiras para tornar as aplicações mais interessantes e mais úteis quanto as aplicações para *desktop*.

Uma das técnicas criadas foi o *Ajax* que segundo Niederauer (2007) "é o uso sistemático de *JavaScript* e *XML* para tornar o navegador mais interativo com o usuário". O *Ajax* utiliza de solicitações assíncronas de informações o que permite fazer solicitações para o servidor sem que seja necessário recarregar a página que está sendo acessada.

O *Ajax*, apesar de existir a bastante tempo, só ganhou fama quando as restrições dos navegadores começaram a ser superadas. Os desenvolvedores comemoraram quando o *Ajax* pode ser utilizado nos principais navegadores e o principal beneficiado foi o usuário final, pois o *Ajax* proporciona uma grande agilidade à atualização de informações web.

Com a utilização do dessa técnica a interação entre o navegador e o servidor não ocorre de maneira direta e como é o *JavaScript* que ativa o *Ajax* o usuário pode permanecer visualizando a página normalmente, enquanto isso o servidor processa a solicitação e envia uma resposta e poderá fazer a atualização de apenas uma parte da página para apresentar a resposta. Assim evitará a retransmissão de informações estáticas melhorando o tráfego da rede e a usabilidade das páginas.

Segundo Niederauer (2007) o principal objetivo do *Ajax* é melhorar a interação do usuário com o servidor, sendo que as páginas devem ser programadas com o foco de não fazer o usuário em vão. Para atingir seus objetivos o *Ajax* utiliza de várias outras tecnologias como *XHTML* e *CSS*, para apresentação, *DOM*, para interação dinâmica, *XML*, para troca de dados, *XHMLHttpRequest*, para chamadas assíncronas e *JavaScript*. Todo o processo do *Ajax* gira em torno do *JavaScript*, pois é por ele que são feitas toda a comunicação entre o usuário e o servidor e também é por onde o *Ajax* será ativado.

No final de 1994, Rasmus Ledorf criou utilitários para sua página pessoal que a monitorava e coletava informações sobre seus visitantes. Conforme o tempo passava mais utilitários foram necessários, então Rasmus escreveu uma implementação utilizando a linguagem C desses utilitários, o que deu origem ao núcleo que ficou conhecido como *PHP/FI*, que é o pacote que deu início ao *PHP* que conhecemos hoje (DALL'OGLIO, 2009).

De acordo com Dall'Oglio (2008) em 1997 foi liberada a segunda versão da implementação C, o *PHP/FI 2.0*, que obteve apoio de milhares de usuários ao redor do mundo, sendo que cerca de 50.000 domínios, o que constituía 1% da *internet*, reportavam sua utilização.

Em 1999 foi desenvolvida a *Zend Engine* cujo objetivo era a melhoria dos pontos críticos apresentados pela linguagem até então. Com base nesta engine e mais uma série de novas características foi criado o *PHP 4.0* em 2000. Na versão 5 que foi lançada oficialmente em 2004, ela passou a suportar Orientação à Objetos de forma consistente, ela foi baseada na *Zend Engine 2. PHP* é um software livre cujo modelo de licenciamento é o *GLP*, o que significa que pode ser instalado em qualquer máquina e para muitos usuários sem que viole alguma lei de direito autoral. Seu suporte é feito por comunidades ao redor do mundo que crescem mais a cada dia, as comunidades são uma das melhores maneiras de obter suporte e trocar experiências (DE MELO, 2007).

De acordo com De Melo (2007) o *PHP*, abreviação de *Hypertext Preprocessor*, é uma linguagem de *script* de código aberto que tem como um de seus principais objetivos a criação de conteúdos dinâmicos para páginas *web* e uma das vantagens dessa linguagem é que ela é executada no lado do servidor, sendo assim o código fonte da aplicação não é apresentada ao internauta que terá apenas acesso ao código *HTML*.

Sistema gerenciador de banco de dados, ou SGBD, é uma coleção de dados que estão inter-relacionados e um conjunto de programas para acessa-los. A coleção de dados, também chamado de banco de dados, contém informações relevantes sobre a

empresa. Um SGBD tem como principal objetivo fornecer uma maneira de recuperar informações que sejam convenientes. Os bancos de dados são amplamente usados, eles formam uma parte essencial das empresas atuais. Seu uso cresceu nas empresas nas últimas quatro décadas do século XX. Um SGBD disponibiliza uma linguagem de definição de dados a linguagem de definição específica, o esquema de banco de dados e também uma linguagem de manipulação de dados que tem a função de expressar as consultas e atualizações de bancos de dados (SILBERSCHATZ, 2006).

Uma linguagem de manipulação de dados permite aos usuários acessar ou manipular dados. Ela pode ser utilizada para recuperação, inserção, exclusão e modificação de informações de informações do banco de dados

Permitir apenas acesso autorizado aos dados e descrever a estrutura dos dados de cada base de dados são tarefas de um *SGBD.* auxilia, através de ferramentas, o usuário a gerenciar uma base de dados, definir os conjuntos de dados que constituirão a mesma, manipular seu conteúdo, gerar relatórios, entre outros (GUIMARÃES, 2003).

A versão original da *SQL* foi desenvolvida pela *IBM* no início da década de 1970. Na sua criação ela tinha o nome de *Sequel*, mas com o passar do tempo seu nome mudou. A linguagem é aceita por muitos produtos e se estabeleceu como a linguagem padrão de banco de dados relacional. Um padrão *SQL*, o *SQL-86*, foi publicado em 1986 pela *American Nacional Standards Institute (ANSI)* e a *International Organization for Standardization (ISO)*, e em 1989 foi publicada um padrão estendido pela *ANSI*, a *SQL-89*. Suas próximas versão foram a *SQL-92*, *SQL-1999* e a mais recente é a *SQL-2003* (SILBERSCHATZ, 2006).

SQL é uma linguagem que especifica exatamente que resultados se deseja obter e não como obter esses resultados. A *SQL* é composta por várias partes sendo elas a linguagem de definição de dados (*DDL*), que permite ao usuário especificar os esquemas das tabelas e linguagem de manipulação de dados (*DML*), permite definir privilégios de acesso e manipulação de tabelas aos usuários (GUIMARÃES, 2003).

O SGDB com suporte para o *SQL* mais utilizado e conhecido a nível mundial com mais de 5 milhões de instalações é o *MySQL* é um sistema de gestão de base de dados relacionais e é *open source*. O *MySQL* surgiu da necessidade que uma equipe tinha de usar algum mecanismo que permitisse conectar as tabelas criadas em *SQL*, mas a ferramenta que iriam utilizar, o mSql, não era rápida o suficiente para as suas necessidades. Em 1996 a companhia suíça *TCX* criou o *MySQL* que atualmente é desenvolvido pela *MySQL AB*. É uma linguagem simples que permite facilmente gravar, alterar e recuperar informações em um site com segurança e velocidade. O Servidor *MySQL* foi desenvolvido para trabalhar com bancos de dados grandes de forma rápida que era o que estava faltando nas soluções existentes e está sendo utilizado de maneira bem-sucedida em ambientes de produção de alta demanda (NEVES, 2005).

2.1 Desenvolvimento

Na fase inicial para o desenvolvimento do sistema marcou-se uma visita a uma empresa que continham os problemas citados anteriormente, onde através de uma reunião levantou-se os requisitos do sistema, como ele deveria funcionar, quais usuários utilizariam e o que eles fariam no sistema e como a empresa em questão trabalhava com seu estoque.

Com a análise dos requisitos coletados possibilitou a criação da modelagem lógica do banco utilizando o programa DBDesigner, a elaboração do diagrama de classes utilizando o software Astah e a elaboração do diagrama de caso de uso utilizando o software Astah.

Através da análise foi possível decidir qual ciclo de vida seria utilizado para o desenvolvimento do sistema em questão, o ciclo escolhido foi o de prototipação, pois foi combinado com o proprietário da empresa que o sistema seriam liberadas versões do sistema para a execução dos testes por etapas.

Após a criação dos diagramas e modelagem do banco iniciou-se o desenvolvimento do sistema.

Na primeira etapa do desenvolvimento criou-se a banco de dados, que continham tabelas referentes a produtos, usuários, movimentação, entre outras, com base na modelagem lógica do mesmo.

Após a criação do banco de dados, desenvolveu-se as telas do sistema levando em consideração os conceitos de ergonomia e usabilidade citadas anteriormente na fundamentação teórica, seguindo um padrão entre elas para que quando o usuário mudar de tela não tenha a sensação de estar acessando um outro sistema. No desenvolvimento do layout do sistema utilizou-se do framework *CSS Bootstrap* que disponibiliza padrões *CSS* prontos que facilita manter o padrão das páginas.

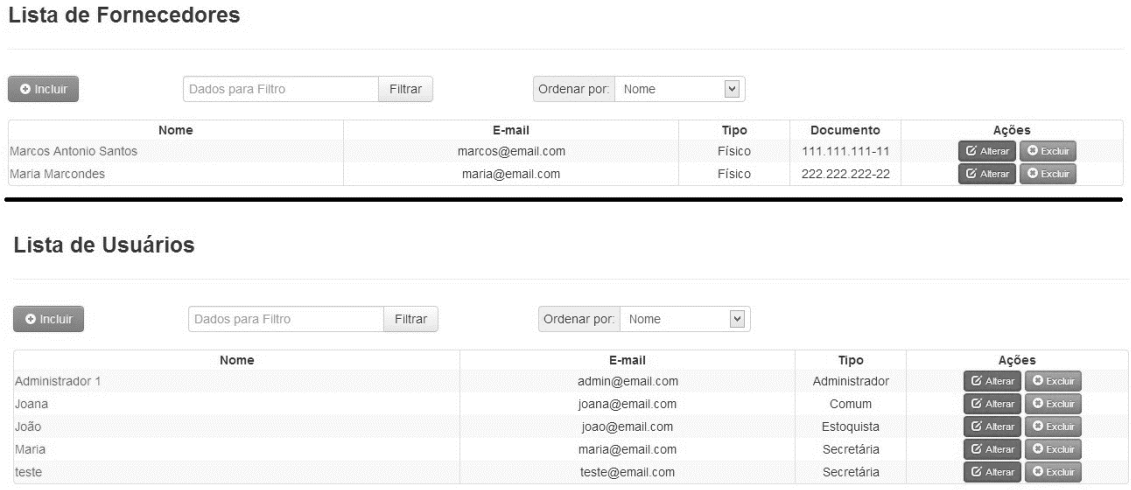


Figura 1. Padrão entre as telas

Foi utilizado o MVC no desenvolvimento do sistema onde as classes ficariam localizadas na pasta *controller* e os arquivos *HTML* na pasta *view*.

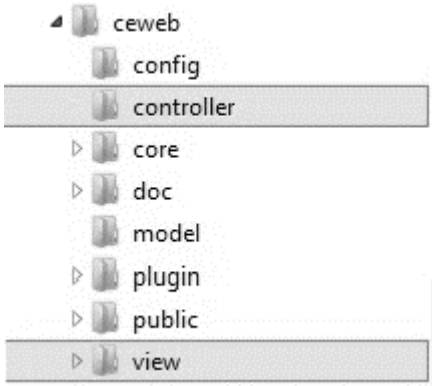


Figura 2. Padrão MVC

Depois da criação do layout e a aprovação do empresário iniciou-se, com a ajuda do Calango Framework, o desenvolvimento das classes que conteriam as funções de *CRUD (Create, Read, Update e Delete)* do sistema, sendo elas a classe de usuários, fornecedores, marcas, categorias, subcategorias e produtos. Essas são as classes que inicialmente disponibilizariam as funções necessárias para o funcionamento.

Com as classes desenvolvidas realizou-se os testes para verificação do funcionamento de suas funções que eram responsáveis pelas operações e as destinadas para a apresentação das informações na tela para o usuário e carregamento de telas em geral.

Os testes que foram realizados pelo empresário envolviam inclusões, alterações, exclusões, verificação de exclusão para dados que estavam presentes em outras tabelas e não poderiam ser excluídos.

Os testes realizados nas funções tinham como objetivo se verificar se estavam realizando suas responsabilidades da maneira correta, já sobre as funções de apresentação tinham como objetivo se estavam apresentando os dados que originavam-se do banco de dados de maneira correta e se não haviam distorções no layout das páginas.

Após realizado os teste foram aplicadas as correções de dados nos formulários, problema nos preços dos produtos ao alterar os mesmos e campos obrigatórios em branco.

Após aplicadas todas as correções prosseguiu-se para a próxima etapa do desenvolvimento onde foi adicionado as funcionalidades que não haviam sido aplicadas na versão anterior. As classes adicionadas ao sistema foram a de movimentação e a responsável pela geração de relatórios.

Na classe de movimentação criou-se funções para alterar a quantidade de produtos em estoque através de entradas e saídas sendo que a classe de produtos não tem esta funcionalidade, ou seja, ela é apenas responsável pelo cadastro dos produtos.

Para a geração dos relatórios desenvolveu-se métodos que fazem a consulta no banco de dados e trazem os dados desejados de acordo com o relatório gerado e utilizando-se da mPDF esses dados se tornam um arquivo pdf com os dados do relatório.

Os relatórios disponibilizados nesta nova versão do sistema foram o relatório geral de produtos, produtos por fornecedor e saída de produtos. Desenvolveu-se uma tela onde seria disponibilizado uma maneira de escolha do relatório desejado e em alguns casos disponibilizando campos para a inserção dos dados necessários.

Após desenvolvidas as classes realizou-se novos testes tendo como objetivo verificar o seu funcionamento, os primeiros testes foram realizados nas movimentações de entrada e saída de produtos, nesses testes verificou-se se os cálculos estavam sendo realizados de maneira correta e se o mesmo permitia realizar uma saída de produtos com valor maior que a quantidade em estoque.

Nos testes dos relatórios foram verificados erros nos formulários para geração dos mesmos e se os relatórios estavam sendo gerados de maneira correta, com estes testes foram verificados que em alguns relatórios deveriam ser alteradas as informações que estavam sendo apresentadas.

Com todas as funcionalidades básicas do sistema desenvolvidas foram adicionadas funções e classe extras. Criou-se uma classe para cadastrar imagens para os produtos para disponibilizar uma maneira de visualizar o produto, adicionou-se na página inicial do sistema um formulário para pesquisa de produtos, produtos com mais saída e produtos esgotados.

Todos os testes foram realizados em servidor remoto na presença do empresário. Após o desenvolvimento ser encerrado foi liberado a versão final para que o empresário pudesse fazer o teste no sistema com todas as funcionalidades apresentadas e apresentar suas opiniões sobre o mesmo e o que poderia ser implementado para próximas versão que tornaria o sistema mais atrativo e funcional para suas necessidades.

3 Resultado

Com o desenvolvimento do sistema obteve-se um aprimoramento no método de consulta dos produtos em estoque, sendo que ele possibilita a consulta dos mesmos sem a necessidade de um funcionário se deslocar até o depósito onde o produto se encontra para fazer a consulta diminuindo assim o tempo que um cliente, por exemplo, esperaria para ter um retorno sobre a disponibilidade do produto desejado.

Com o relatório de saída de produtos pode-se obter uma frequência de venda dos produtos, com esses dados obtidos pode-se realizar uma previsão de saída de cada produto e quantos produtos serão necessários em estoque até a próxima data de compra.

Com a utilização do relatório por fornecedor permite fazer uma melhor verificação de quais produtos necessitam ser comprados de cada fornecedor. O relatório geral permite uma visão de como o estoque se encontra no momento.

Com a execução deste trabalho obteve-se resultados que ajudaram a melhorar o conhecimento sobre as tecnologias utilizadas no desenvolvimento do sistema tanto nas suas funcionalidades quando na sua teoria e histórico de como e porque ela foi criada.

Obteve-se também conhecimento sobre estoque e de como deveriam ser executados os processos dentro de uma empresa para que o sistema torna-se as rotinas diárias mais fáceis e rápidas de serem executadas.

4 Considerações finais

Futuramente pretende-se desenvolver uma nova função ao sistema, a funcionalidade de vendas, que através dele será disponível fornecer um relatório mais detalhado sobre a saída de produtos. Serão incluídos novos cadastros ao sistema, sendo eles cadastro de clientes, de depósitos.

Pretende-se também a implementação de gráficos para uma melhor visualização das informações importantes para a tomada de decisões de uma empresa, por exemplo gráficos de frequência de saída de produtos, gráficos de saída de produtos por mês, entre outras implementações.

O sistemas como foi entregue possui funcionalidades que permitem, se o empresário desejar, que o mesmo possa tornar-se um sistema e-commerce com a inclusão de novas funcionalidades especificas para o que for pedido.

Referências

Cybis, Walter, Adriana Holtz Betoil, Richard Faust. – Ergonomia e Usabilidade: conhecimentos, métodos e aplicações – 2ª Edição – São Paulo: Novatec, 2010.

Dall'Oglio, Pablo – PHP – Programando com Orientação a Objetos – 2ª Edição – São Paulo: Novatec, 2011.

De Melo, Alexandre Altair, Nascimento, Mauricio G. F. – PHP Profissional – São Paulo: Novatec, 2007.

Erich Gamma, Ralph Johnson, Richard Helm, John Vlissides - Padrões de Projeto: Soluções reutilizáveis de software orientado a objetos – São Paulo: Bookman, 2006.

Guimarães, Célio Cardoso, 1942 – Fundamentos de bancos de dados: modelagem, projeto e linguagem SQL – Campinas: São Paulo, 2003.

Neves, Pedro M. C., Ruas, Rui P. F. – Guia Prático do MySQL – Portugal: Centro Atlântico, 2005.

Niederauer, Juliano – Web Interativa com AJAX e PHP – São Paulo: Novatec Editora, 2007.

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

Shitsuka, Rabbith I. C.; Shitsuka, Caleb D. W. M.; Shitsuka, Ricardo; Shitsuka, Dorlivete M. – Sistemas de Informação: Um Enfoque Computacional – Rio de Janeiro: Ciência Moderna Ltda, 2005.

Silberschatz, Abraham, Henry F. Korth, S. Sudarshan; tradução de Daniel Vieira - Sistema de Banco de Dados– Rio de Janeiro: Elsevier, 2006 – 9ª reimpressão.

Silva, Mauricio Samy – HTML5 – São Paulo: Novatec, 2011.

Silva, Mauricio Samy – Jquery – A Biblioteca do Programador JavaScript – 2ª Edição – São Paulo: Novatec, 2010.

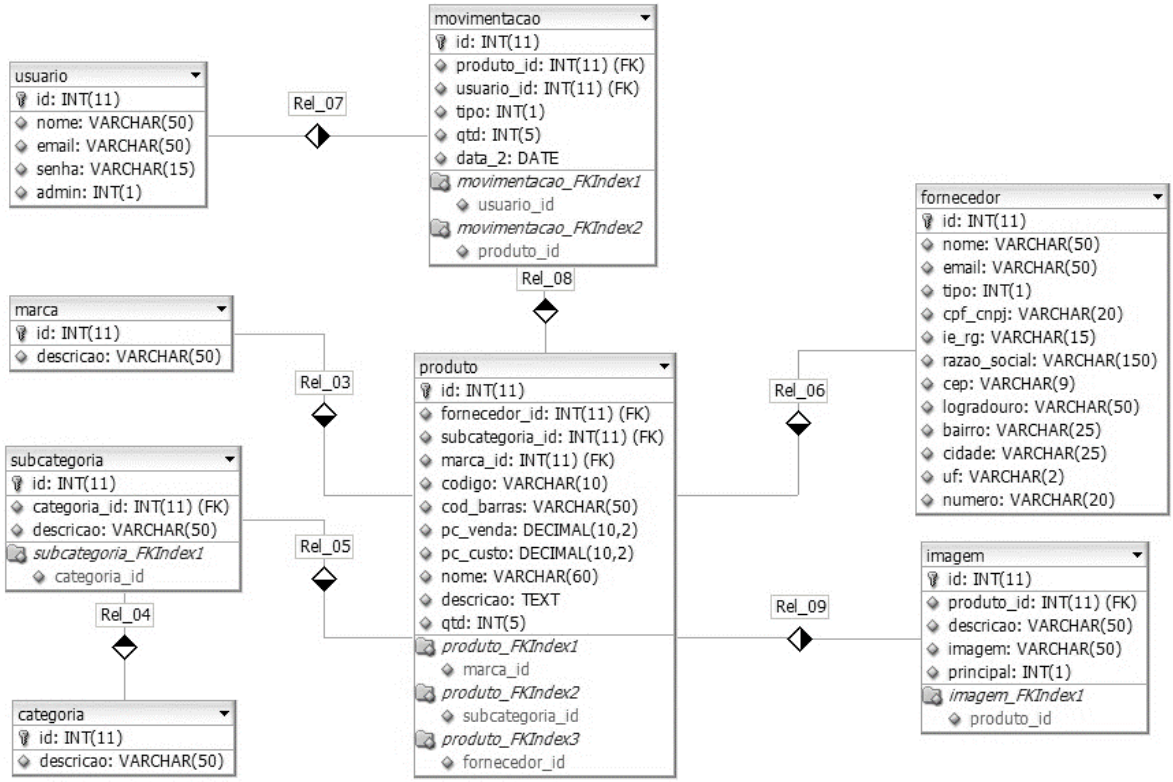
Silva, Mauricio Samy – JavaScript - Guia do Programador – São Paulo: Novatec, 2010.

Silva, Maurício Samy –Construindo sites com CSS e (X)HTML: sites controlados por folhas de estilo em cascata – São Paulo: Novatec, 2008.

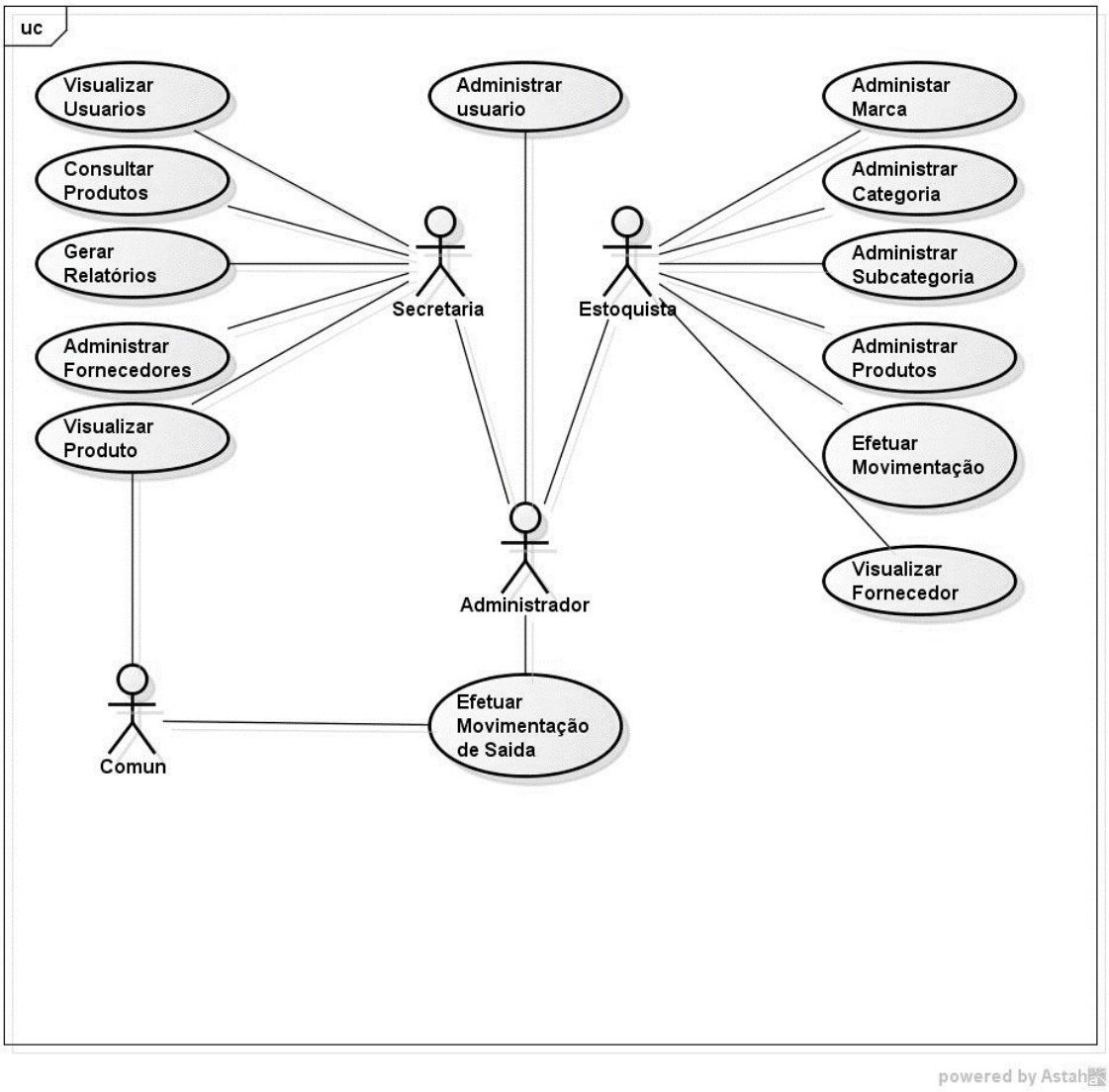
Viana Vargas, Ricardo – Gerenciamento de Projetos: estabelecendo diferenciais competitivos – 6ª Edição – Rio de Janeiro: Brasport, 2005.

5 Apêndices

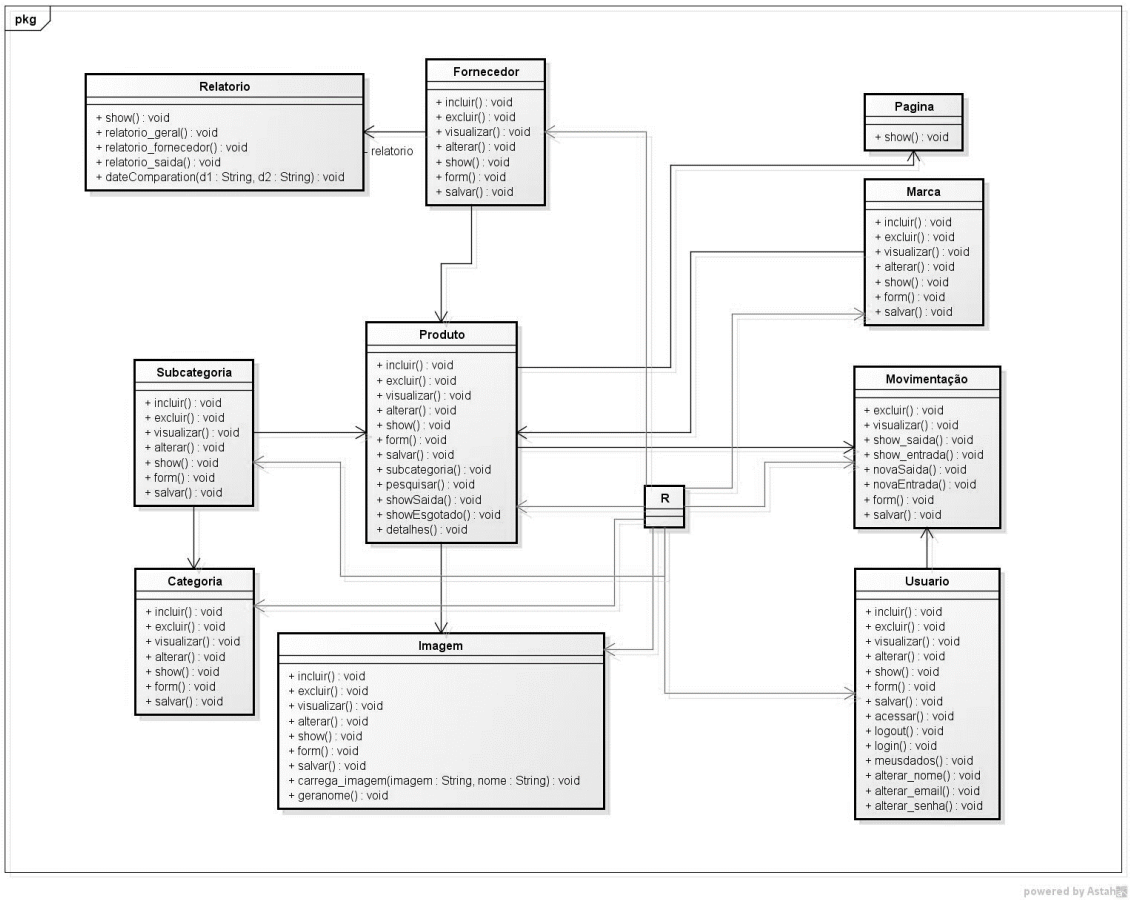
5.1 Modelagem do Banco de Dados



5.2 Diagrama de Caso de Uso



5.3 Diagrama de Classes



5.4 Diagrama de Sequência