RESIGeT On-Line
Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica
Vol. 02 - Nr. 01 - 2012
Uni-FACEF CENTRO UNIVERSITÁRIO DE FRANCA

Software de Colméia Eletrônica para Controle de Bares, Lanchonetes e afins "eColméia"

CORREIA, Pedro Paulo[1]
peezzim@gmail.com

SILVA, Matheus Theodoro[1]
matheusth.si@hotmail.com

TANAKA, Luís Carlos[1]
tanaka@tanaka.pro.br

Jaqueline Brigladori Pugliesi[2]
jbpugliesi@gmail.com

Resumo
O presente artigo apresenta o projeto de desenvolvimento de um software de Colméia Eletrônica para controle de Bares, Lanchonetes e afins. Por meio de análise bibliográfica e documental, busca-se a partir de dados secundários, apresentar uma visão geral da linguagem de programação orientada a objetos, da ferramenta de desenvolvimento e do banco de dados utilizados no software em questão. Este artigo disserta brevemente sobre o funcionamento do software proposto, apresentando as telas do programa desenvolvido e com isso, apresenta os próximos passos a serem implementados oportunamente.

Palavras chave: Software, lanchonete, bar, colmeia.

Abstract
This paper presents the project of a software development Beehive Electronics for control bars and snack bar and the like. Through literature review and documentary seeks based on secondary data, present an overview of the programming language object-oriented, the development tool and database used to develop the software in question This article, spoken briefly about the operation of the proposed software screens showing the program developed and thus sets out the next steps to be implemented in due course.

Keywords: Software, Snak Bar, Bar, Hive.

1 Alunos regularmente matriculados no 4º semestre do curso de Sistemas de Informação do Centro Universitário de Franca – Uni-FACEF.
2 Docente do curso de Sistemas de Informação do Centro Universitário de Franca – Uni-FACEF

INTRODUÇÃO

O objetivo principal deste artigo é apresentar o desenvolvimento de um software de colméia eletrônica para o controle de lanchonetes, bares e afins.

Neste trabalho não se tem a pretensão de esgotar, no todo, o assunto objeto do tema, mas contribuir para a divulgação dos resultados das competências adquiridas nas disciplinas da primeira turma do curso de Sistema de Informação do Centro Universitário de Franca – Uni-FACEF, no ano de 2011.

Os conteúdos pesquisados foram complemento do aprendizado em sala de aula, aplicando-se na prática os mais diversos temas até então abordados na teoria da academia, proporcionando um grande incremento de experiências e aperfeiçoamento de conhecimentos adquiridos.

A motivação para o desenvolvimento deste software se deu pela percepção que os pequenos negócios, na maioria um pequeno empreendimento familiar, utiliza-se da chamada “colméia”, pequenas caixas de madeira imitando uma colméia de abelhas, para a guarda provisória das comandas descrevendo os pedidos das mesas. A proposta então seria de um software de controle para transformar esta realidade em outra informatizada e mais eficiente, sem prejuízo para o negócio.

O software proposto está em sua fase “embrionária”, sugerindo assim que diversas modificações se farão necessárias até a conclusão do projeto, porém, deixa um legado importante em relação ao curso e na formação profissional do discente.

Por fim, espera-se que o fato de não se ter um “produto finalizado” seja uma motivação adicional para a continuidade do projeto e engrandecimento do conhecimento dos alunos envolvidos.

1. SOFTWARE

O software ou programa de computador apresenta diversas definições, como a seguinte, presente em uma licença de software da IBM:

> “... O termo "Programa" significa o programa original e todas as cópias completas ou parciais do mesmo. Um Programa consiste em instruções legíveis por máquina, seus componentes, dados, conteúdo audiovisual (tal como imagens, texto, gravações ou figuras) e materiais licenciados relacionados.”

De acordo com esta definição, em geral adotada pela indústria de software, um software é mais do que as instruções interpretáveis por uma máquina e tem por objetivo descrever tudo que é utilizado e produzido para o usuário ou sua máquina, também é parte do software.

Para Fernandes (2003), o software é um “artefato virtual, incapaz de realizar trabalho a menos que exista uma máquina que carregue e interprete as instruções e informações contidas no mesmo”, destacando a necessidade de uma máquina para compor um sistema informatizado.

1.1 Engenharia de software

A engenharia de software é uma atividade industrial de produção de software. Fernandes (2003) a define como sendo uma disciplina do conhecimento humano que tem por objetivo definir e exercitar processos: humanos atuando como máquinas; métodos: planos de processos, ferramentas e ambientes: máquinas apoiando processos e métodos; para construção de software que satisfaça necessidades, dentro de prazos e custos previsíveis.

Para Fernandes (2003), "o desenvolvimento de software está no cerne de uma relação humana de troca de planos, posses, desejos e necessidades entre três categorias de agentes coletivos: os que usam, os que adquirem e os que produzem software", conforme ilustrado na Figura 01.

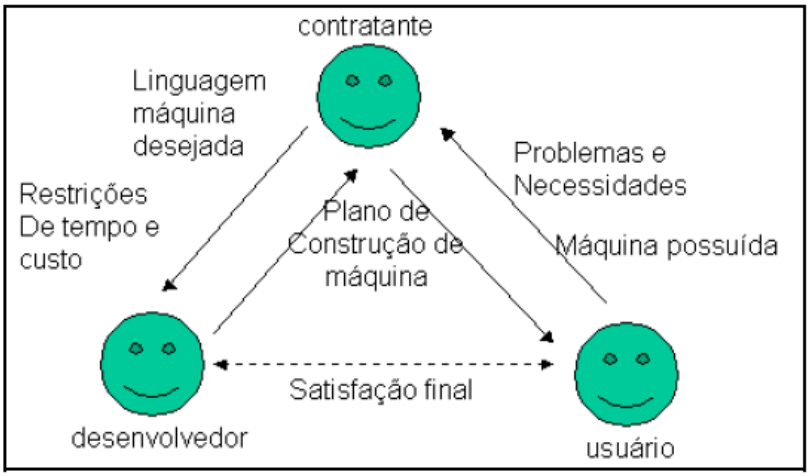

Figura 01: Relação entre participantes do software
Fonte: (FERNANDES, 2003)

O corpo de conhecimento da engenharia de software é estruturado em áreas

de atividades, destacando-se entre outros: Requisitos de software: aumentar a precisão e controlar a interação entre a máquina e o usuário; Design de software: particionar e organizar o plano de construção do software; Construção de software: definir instruções e descrições para cada um dos planos de máquina; Testes e Qualidade de software: para atestar que software satisfaz a necessidades propostas. (FERNANDES, 2003)

Parreiras e Oliveira aput DeSouza (2004) consideram a engenharia de software como sendo um domínio do conhecimento "no qual os fatores de sucesso estão relacionados com a experiência das pessoas envolvidas nas fases de projeto, construção, teste e implantação", conferindo a importância destas fases no desenvolvimento de softwares.

1.2 Desenvolvedor

Na visão de Fernandes (2003), o desenvolvedor de software é um agente coletivo, responsável por criar um plano de construção de máquina (software) que esteja dentro das condições estabelecidas pelo cliente, através da de uma definição de linguagem necessária ao usuário (LNU), com gramática (sintaxe) e lógica (semântica) bem definidas.

1.3 Cliente

O consumidor do software, usualmente chamado de cliente, é uma entidade que adquire uma cópia de um software, fornecida por um agente que será chamado de desenvolvedor, através de algum processo de troca, que pode envolver entre outras coisas, dinheiro, bens, ou redes de conhecimento. (FERNANDES, 2003)

1.4 Usuário

Quem usa um software em geral é chamado de usuário, podendo em muitos casos também ser o cliente.

Referente a senhas, Tanenbaum destaca:

> “Alguns sistemas [...] exigem que os usuários alterem suas senhas regularmente para limitar o prejuízo se uma senha for descoberta. Nesse caso, o compromisso é que, se tiverem de mudar suas senhas muito freqüentemente, os usuários desprezarão aquelas senhas boas e escolherão as fáceis. Se forem impedidos de escolher as fáceis,

> eles as esquecerão e as escreverão em blocos de notas com adesivos para colarem em seus monitores, o que abre um grande buraco na própria segurança". (TANENBAUM, 2003, p.450)

2. LINGUAGEM DE PROGRAMAÇÃO

2.1 Java

A linguagem de programação Java foi criada em 1991 por James Gosling, ela iniciou-se como parte do projeto Green da Sun Microsystems. Existem diversas linguagens de programação e cada uma tem seu espaço e seu melhor uso. Optar pelo o uso do Java é interessante em aplicações que virão a crescer, em que a legibilidade do código é importante, onde tem-se muita conectividade e se há muitas plataformas, ambientes e sistemas operacionais, heterogêneas como Linux, Unix, OSX e Windows misturados. (CAELUM, 2011, p.7)

A linguagem de programação Java é uma linguagem de alto-nível com as seguintes características: Simples: O aprendizado da linguagem pode ser feito em um curto período de tempo; Orientada a objetos: Desde o início do seu desenvolvimento esta linguagem foi projetada para ser orientada a objetos; Familiar: É muito familiar para os programadores C/C++; Robusta: Ela foi pensada para o desenvolvimento de softwares confiáveis, provendo verificações tanto em tempo de execução quanto compilação, o coletor de lixo responsabiliza-se pela limpeza da memória quando houver necessidade; Segura: Aplicações Java são executadas em ambiente próprio (JRE) o que inviabiliza a intrusão de código malicioso; Portável: Programas desenvolvidos nesta linguagem podem ser executados em praticamente qualquer máquina desde que esta possua o JRE instalado. (EIJE, 2011, p.3-4)

A máquina virtual java (JVM) é uma máquina imaginária que emula uma aplicação em uma máquina real. É a JVM que permite a portabilidade do código Java, isto ocorre porque todo código Java é compilada para um formato intermediário, bytecode, este formato é então interpretado pela JVM. (EIJE, 2011, p.4).

Na linguagem de programação Java a responsabilidade pela gerência da memória é do Coletor de lixo (Garbage Collector), desta forma, programadores Java

ficam livres da preocupação de alocação e desalocação da memória. O Coletor de lixo é um processo que roda em segundo plano e é responsável pela liberação de memória alocada por variáveis que não mais serão utilizadas pela aplicação.

As fases pelo qual passa um programa Java estão relacionadas conforme ilustrado na Figura 02: Criação do código fonte (Programa.java); Compilação do código fonte e geração do bytecode (Programa.class); Interpretação do bytecode pela máquina virtual; Conversão do bytecode em linguagem de máquina. (EIJE, 2011, p.5).

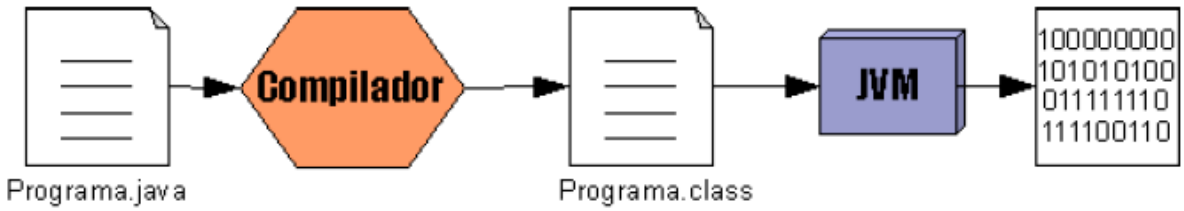

Figura 02: Fases do Programa Java
Fonte: Adaptado de EIJE, Albert (2011. p.5).

Com uma linguagem orientada a objetos e madura como o Java, fica mais fácil e rápido fazer alterações no sistema, desde que se siga as boas práticas e recomendações sobre design orientado a objetos.

A linguagem de programação Java e o sistema de execução Java (*run-time system*) que a acompanha foram projetados para que um programa pudesse ser escrito e compilado uma vez e então transportado pela Internet na forma binária e executado em qualquer máquina que suportasse java. A segurança faz parte do projeto do Java desde o início. (TANENBAUM, 2003, p.481)

3. DADOS

3.1 Estrutura de Dados

Uma estrutura de dados mantém os dados organizados seguindo alguma

lógica e disponibilizam operações para o usuário manipular os dados. (CAELUM, 2011, p.4). Ainda segundo Caelum (2011. P.4), é importante, quando programar, não misturar dado e estrutura de dados. Um dado é uma informação armazenada e estrutura de dados é o que administra os dados. O ideal é que a estrutura de dados seja o mais independente possível dos dados que ela irá armazenar.

Coleções são estruturas de dados, ou seja, é um conjunto de elementos contidos em uma única estrutura – em Java um objeto – cuja função é oferecer meios de armazenar, disponibilizar, remover, localizar e percorrer o seu conteúdo. Os tipos de coleções mais comuns são: Vetor: É formado por um grupo de elementos acessados através do seu índice; Pilha: Estrutura de dados onde o último elemento a ser inserido na coleção é o primeiro a ser retirado, baseado no princípio LIFO, “Last in, first out”; Fila: Coleção onde a ordem de inserção representa a ordem de saída dos elementos, baseado no princípio FIFO, “First in, first out”; Árvores: Estrutura de dados que garante a ordenação dos elementos que a compõe. (CAELUM, 2011, p.4)

Um array é um objeto que armazena um número pré-definido de elementos, isto é, o seu tamanho é definido no momento da sua construção. Podem ser unidimensionais: estruturas de dados bastante simples ou multi-dimensionais: Uma estrutura um pouco mais complexa que permitem a construção de estruturas de dados mais ricas. (CAELUM, 2011, p.8)

3.1 Banco de Dados e SQL

Na definição de Silberscatz (2006. p.1), um “sistemas de gerenciamento de banco de dados (DBMS) é uma coleção de dados inter relacionados e um conjunto de programas para acessar esses dados”.

As coleções de dados chamadas de Banco de Dados são projetadas para gerenciar grandes blocos de informações. O principal objetivo de um DBMS é fornecer uma maneira de recuperar informações de banco de dados que seja conveniente e eficiente ao mesmo tempo. Além disso, o sistema de banco de dados precisa garantir a segurança das informações armazenadas. (SILBERSCATZ, 2006, p.1).

O banco de dados é baseado no modelo relacional e usa um conjunto de tabelas para representar os dados e as relações entre eles, inclui também uma DML (Linguagem de Manipulação de Dados), que permite aos usuários acessar e manipular dados e uma DDL (Linguagem de Definição de Dados), usada para especificar as propriedades adicionais dos dados, formando assim uma única linguagem de banco de dados denominada SQL (*Structured Query Language*). (SILBERSCATZ, 2006, p.6)

Embora SQL refere-se a uma "linguagem de consulta", ela pode fazer muito mais do que simplesmente consultar um banco de dados. Ela pode definir a estrutura de dados, modificar dados no banco de dados e especificar restrições de segurança, sendo, portanto a linguagem padrão de banco de dados relacional. (SILBERSCATZ, 2006, p.51).

Uma das principais vantagens da linguagem SQL é permitir a inclusão de programas que ficam armazenados no próprio banco como os *triggers* e as *stored procedures*, que podem ser reutilizados por qualquer aplicação que acesse este banco, e não apenas submeter instruções da linguagem SQL como: *update, select, insert* e *delete*.

3.1 MySQL

O MySQL, Figura 03, é um sistema de gerenciamento de banco de dados (SGBD), que utiliza a linguagem SQL como interface. É um Software Livre com base na licença GPL em estrutura cliente/servidor. Consiste de um servidor SQL multitarefa que suporta acessos diferentes, diversos programas clientes e bibliotecas.

4. SOFTWARE eColméia

O sistema tem com o intuito de dar suporte para a parte operacional de pequenas lanchonetes e bares tornando acessível a estes comércios a possibilidade de utilizarem um sistema que permita o cadastro e gerenciamento de produtos, departamentos, garçons e usuários. Algumas telas do software eColméia são

ilustradas na Figura 03.

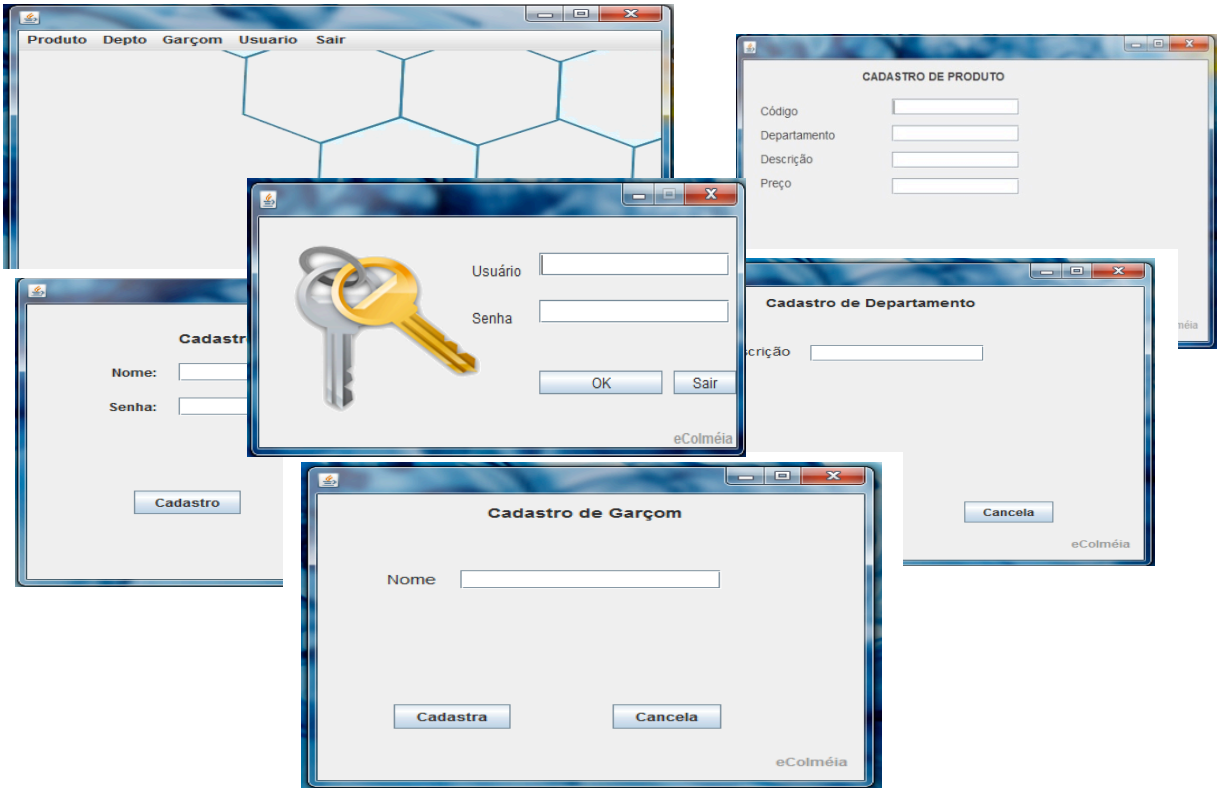

Figura 03: Telas do Software eColméia

Para que fosse possível alcançar os objetivos propostos, foram priorizadas ferramentas de softwares livres, sendo utilizada a IDE (*Interface Development Environment* – Interface de Desenvolvimento Integrado), o Netbeans na versão 6.9.1, conjunto de desenvolvimento da linguagem Java para a criação da aplicação. A linguagem Java segundo Gonçalves (2006. p.1) é utilizada por grandes bancos e empresas, pois fornece extrema segurança, para os que desejam trafegar uma grande quantidade de dados e necessitam estabilidade e portabilidade.

Para dar suporte, foi utilizado o banco de dados MySQL e aplicação construtor MySQL WorkBench 5.2 CE para criação e gerenciamento do banco de dados, tabelas e acessos. Dessa forma, preocupou-se com os custos referentes ao sistema de banco de dados, visto que o MySQL possui o licenciamento GNU, ou seja, é gratuito. A imagem da Figura 04 ilustra o Modelo de Entidade e Relacionamento (MER) do banco de dados utilizados no desenvolvimento do software.

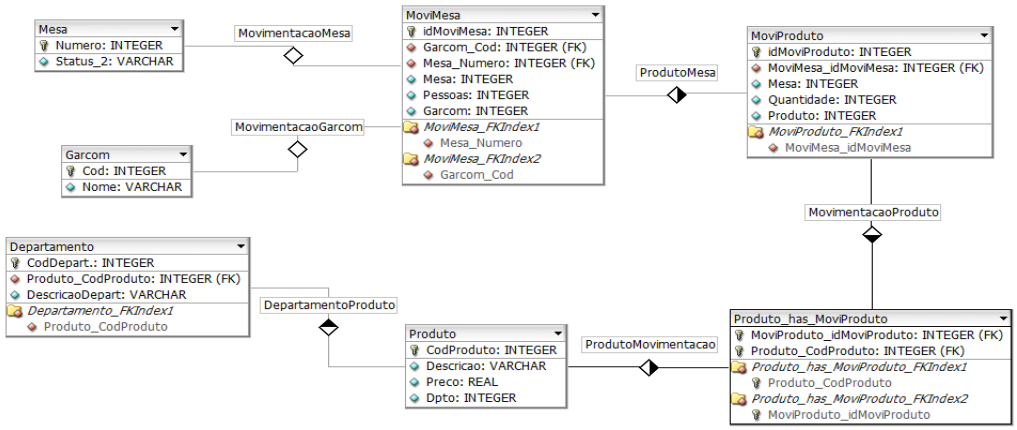

Figura 04: MER no Banco de Dados

O driver JDBC – *Java Data Base Conector* (Conector de Banco de Dados Java em português), foi necessário para permitir que todas aplicações Java acessem o banco de dados MySQL e seus dados e tabelas.

4.1 Padrões de Desenvolvimento

A aplicação foi construída utilizando o padrão MVC (*Model – View – Controller* ou Modelo – Visualização – Controle), ilustrado na Figura 05, que conforme Pires (2011), possibilita o mapeamento da entrada, o processamento e a saída de dados, além da interação com o usuário em camadas.

> A arquitetura MVC fornece uma maneira de dividir a funcionalidade envolvida na manutenção e apresentação dos dados de uma aplicação. A arquitetura MVC não é nova e foi originalmente desenvolvida para mapear as tarefas tradicionais de entrada, processamento e saída para o modelo de interação com o usuário. Usando o padrão MVC fica fácil mapear esses conceitos no domínio de aplicações Web multicamadas. (MACORATTI, 2011)

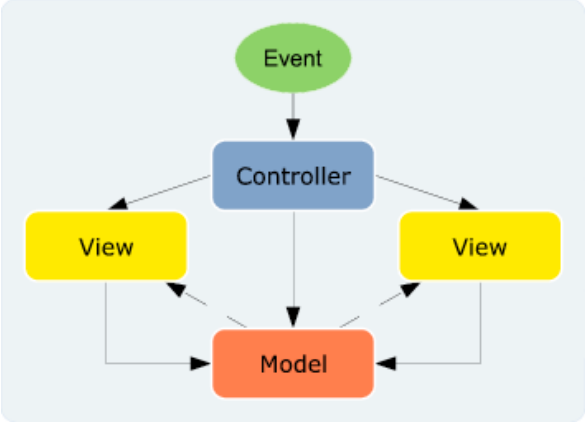

Figura 05: Padrão MVC
Fonte: http://fabriciosanchez.com.br/site/?p=1014

Ainda de acordo com Pires (2001), as camadas do padrão de desenvolvimento e suas respectivas características são: *Model*: Representa os dados da aplicação e como a mesma irá se comportar com tais dados. A *model* mantém o estado persistente do negócio e permite ao controlador a capacidade de acessar as funcionalidades da aplicação encapsuladas pelo próprio modelo; *View:* Um componente de visualização *renderiza* o conteúdo de uma parte particular do modelo e encaminha para o controlador as ações do usuário. Acessa também os dados do modelo via controlador e define como esses dados devem ser apresentados; *Control*: É a camada intermediária entre o *view* e *model*, cabendo a ela interpretar o usuário e acessar a *model*. Há normalmente um controlador para cada conjunto de funcionalidades relacionadas.

Outro padrão de projeto utilizado foi o padrão *Data Access Object* (DAO – Objeto de Acesso a Dados), que tem como essência agir entre o componente e a fonte de dados, de tal forma que os códigos ficam separados das camadas de acesso a dados, facilitando o seu entendimento, conforme ilustrado na Figura 06. (SUN, 2011).

Figura 06: Data Access Object - DAO
Fonte: http://reginaldojr.wordpress.com/tag/padrao-de-projeto/

O objetivo do DAO é separar das regras da aplicação a tecnologia de persistência de dados. Neste tipo de aplicação nenhuma regra deve estar no SGDB. Ele passa a ser apenas um repositório de dados, ou seja, um local onde os dados são armazenados e consultados. *Triggers, procedures, functions, roles e views* são dispensáveis neste contexto. (MACORATTI, 2011).

Por fim, também foi utilizado o padrão FACTORY que, segundo a definição de Destro (2004), possibilita fazer alterações no software sem que o cliente perceba. A Figura 07 ilustra o padrão FACTORY.

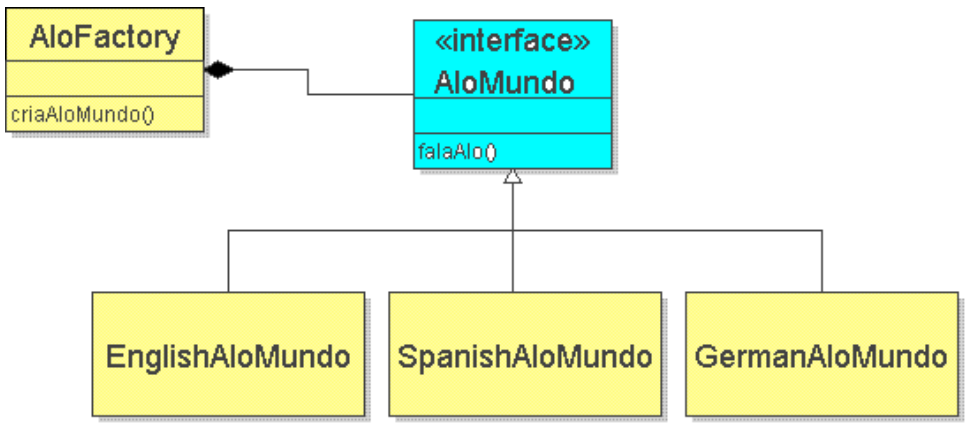

Figura 07: Factory
Fonte: http://reginaldojr.wordpress.com/tag/padrao-de-projeto/

O padrão FACTORY fornece uma interface para a criação de famílias de objetos correlatos ou dependentes sem a necessidade de especificar a classe concreta destes objetos. Este padrão é útil quando você precisa criar objetos dinamicamente sem conhecer a classe de implementação, somente sua interface: o padrão factory estabelece uma forma de desenvolver objetos que são responsáveis

pela criação de outros objetos. (MACORATTI, 2011).

CONSIDERAÇÕES FINAIS

Este trabalho apresenta uma pesquisa de natureza exploratória realizada por extensa procura em dados secundários por meio de busca em referencial bibliográfico e documental para complementar os conhecimentos adquiridos nas disciplinas do curso de Sistema de Informação.

Nesta etapa foram realizadas ainda entrevistas informais com especialistas e profissionais de empresas da área de lanchonetes e bares, visando compreender mais profundamente o tema, bem para a análise de requisitos para o desenvolvimento do software.

Espera-se que o fato de não se ter um “produto finalizado” seja somente mais uma motivação para a continuidade do projeto, que visa sempre o engrandecimento do conhecimento dos alunos envolvidos.

REFERÊNCIAS BIBLIOGRÁFICAS

ANSELMO, Fernando. ***Aplicando Lógica Orientada a Objeto em Java***. 2.ed. atual. e ampl. Florianópolis: Visual Books. 2005.

CAELUM, Equipe. ***Curso FJ-11: Java e Orientação a Objetos***. Disponível em: www.caelum.com.br/apostilas. Acessado em 21/11/2011.

CAELUM, Equipe. ***Curso CS-14: Algoritmo e Estrutura de Dados em Java***. Disponível em: www.caelum.com.br/apostilas. Acessado em 21/11/2011.

DESTRO, D. Implementando Design Patterns com Java. Disponível em: http://www.guj.com.br/articles/137. 2004. Acessado em 21/11/2011.

EIJE, Albert. ***Curso Java Starter - Módulo 01: Introdução ao Java***. T2TI TECNOLOGIA DA INFORMACAO LTDA. Disponível em: www.t2ti.com. Acessado em 21/11/2011.

EIJE, Albert. ***Curso Java Starter - Módulo 03: Introdução ao Java***. T2TI TECNOLOGIA DA INFORMACAO LTDA. Disponível em: www.t2ti.com. Acessado em 21/11/2011.

FERNANDES, Jorge Henrique Cabral. ***Qual a Prática do Desenvolvimento de Software?*** Revista Ciência e Cultura. Vol 55. N 2. Abril / Maio / Junho de 2003. Páginas 29 a 33. Editora da Sociedade Brasileira para o Progresso da Ciência. Disponível em: http://www.dimap.ufrn.br/~jorge/textos/papers/PraticaDeSoftware.pdf.

GONÇALVES, Edson. ***Dominando NetBeans***. Rio de Janeiro: Editora Ciência Moderna. 2006.

IBM. ***Termo de Licença de Softwre***.

MACORATTI, José Carlos. VB.NET - Design Patterns - Factory. Disponível em: http://www.macoratti.net/vbn5_dp1.htm. 2011. Acessado em: 21/11/2011.

MACORATTI, José Carlos. Padrões de Projeto : O modelo MVC - Model View Controller. Disponível em: http://www.macoratti.net/vbn_mvc.htm. 2011. Acessado em 21/11/2011.

SUN. Sun Developer Network, Núcleo Catalogo de Padrões J2EE. Disponível em : http://java.sun.com/blueprints/corej2eepatterns/Patterns/DataAccessObject.html 2010. Acessado em 21/11/2011.

SILBERSCATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S. ***Sistema de Banco de Dados***. Tradução Daniel Vieira. Rio de Janeiro: Elsevier. 2006.

PARREIRAS, F. S., OLIVEIRA, G. S. ***Análise comparativa de processos de desenvolvimento de software sob a luz da gestão do conhecimento: um estudo de caso de empresas mineiras.*** In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 3, 2004, Brasília. Anais... , 2004. Disponível em: http://www.fernando.parreiras.nom.br/publicacoes/WGC_Parreiras04.pdf.

PIRES, D.F. MVC: Model – View - Controller. 2011.

TANENBAUM, Andrew S. ***Sistemas operacionais modernos***. Tradução Ronaldo A.L. Gonçalves, Luís A. Consularo. Revisão técnica Regina Borges de Araújo. 2. ed. São Paulo: Pearson Prentice Hall. 2003.