CENTRO UNIVERSITÁRIO UNIFACVEST

CURSO DE CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO

CARLOS GUILHERME COELHO

TCS: Sistema de Geração de Avaliações Automatizado

LAGES

2014

CARLOS GUILHERME COELHO

TCS:Sistema de Geração de Avaliações Automatizado

Projeto apresentado à banca examinadora do
Trabalho de conclusão de curso de ciências da
Computação para análise e aprovação

LAGES

2014

CARLOS GUILHERME COELHO

Sistema de Geração de Avaliações Automatizado

Trabalho de Conclusão do Curso de Ciência da Computação apresentado ao Centro Universitário UNIFACVEST como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação.
Prof. MSc. Márcio José Sembay

Lages, SC ____/____/2014. Nota_____ _____

_____

_____

_____

LAGES

2014

Resumo

Este trabalho apresenta um sistema gerador de avaliações, com o objetivo de automatizar e padronizar a geração de provas. Para atingir tal meta foram realizadas pesquisas bibliográficas referentes a padrões em sistemas computacionais, orientação a objetos e persistência em banco de dados. O desenvolvimento de aplicações que automatizem diversos processos na vida dos usuários é cada vez maior, variando de pequenos aplicativos de previsão do tempo até sistemas de gestão financeira. A padronização das avaliações facilita a construção, reutilização, dificuldade e aplicação de um teste pelo professor, e norteia o aluno à continuidade e formato da avaliação, pois uma vez conhecido o padrão, não haverá surpresa ao realizar uma avaliação que segue a mesma metodologia. Com o sistema gerador de avaliação, o professor padrão de resposta necessita apenas cadastrar as questões referentes a sua disciplina e depois gerar uma avaliação com quantidade, nível e formato de questões que desejar.

Palavras chave: Gerador de avaliações, Sistemas computacionais, padrão de resposta.

Abstract

This paper presents a system for generating tests aiming, to standardize and automate the generation of tests. To achieve this goal literature searches relating to standards in computing systems, object orientation and persistence in the database were performed. The development of applications that automate various processes in the life of users is increasing, ranging from small applications of weather forecasting to financial management systems. The standardization of ratings facilitates the construction, reuse, implementation and difficulty of a test by the teacher guides the student and the continuity and shape of the test, once known as the standard, there will be no surprise to an assessment that follows the same methodology. With the generator rating system, the response pattern teacher needs only registering concerns about their discipline and then generate an evaluation with quantity, level and format of questions that you want.

Keywords : Generator reviews, computer systems, response pattern.

Resume

Este trabajo presenta un sistema para generar pruebas con el objetivo de estandarizar y automatizar la generación de pruebas. Para lograr este objetivo búsquedas bibliográficas relativas a las normas en el cálculo se realizaron los sistemas, la orientación a objetos y la persistencia en la base de datos . El desarrollo de aplicaciones que automatizan diferentes procesos de la vida de los usuarios es cada vez mayor, que van desde pequeñas aplicaciones de la predicción del tiempo para los sistemas de gestión financiera. La estandarización de las clasificaciones facilita la construcción, la reutilización, y la dificultad para la aplicación de una prueba por el profesor guía al estudiante y la continuidad y forma de la revisión, una vez conocido como el estándar, no será ninguna sorpresa para una evaluación que sigue la misma metodología. Con el sistema de calificación del generador, el maestro patrón de respuesta sólo necesita el registro de las preocupaciones sobre su disciplina y luego generar una evaluación con la cantidad, el nivel y el formato de las preguntas que desee.

Palabras clave: opiniones electrógenos, sistemas informáticos, patrón de respuesta

LISTA DE ABREVIATURAS

PDF - *Portable Document Format*

4GL's - Linguagens de 4° geração

DLL - *Dynamic-link library*

CLR - *Commom Language Runtime*

CTS - *Common Type Specification*

SQL - *Structured Query Language*

T-SQL - *Transact- Structured Query Language*

POO - Programação Orientada a Objetos

TCS - *Test Creator System*

LISTA DE FIGURAS

SUMÁRIO

I. INTRODUÇÃO

Desde a criação dos primeiros computadores na década de 40, os softwares criados tinham como objetivo, ajudar o usuário a completar uma tarefa mais rapidamente. Quando uma nova tecnologia é bem-sucedida na resolução de um problema, rapidamente se torna uma tendência que será utilizada pela maioria dos desenvolvedores na criação de software e se torna o próximo nível a ser modificado pela própria tecnologia (PRESSMAN, 2010). As tecnologias evoluíram e os problemas que devem ser solucionados seguem um ritmo de evolução ainda maior. Cada ano dezenas de milhares de novos usuários se lançam em busca de sistemas que ofereçam soluções rápidas e objetivas para os mais diversos problemas.

O senso comum ensina "o ser humano teme o desconhecido", partindo desse ponto. Observamos que a metodologia de avaliação feita por um professor se difere dos demais professores e assim respectivamente, porque após criar uma zona de segurança o sistema será o mesmo sem mudanças. Porem um acadêmico que tem diversas aulas, sobre variados temas durante sua graduação tem que se adequar a cada sistema, dificultando assim sua preparação para uma avaliação.

A padronização no formato de provas cria uma segurança ao aluno que já sabe o que esperar sobre a avaliação e ajuda ao professor manter o equilíbrio entre dificuldade e conteúdo cobrado na avaliação.

1.1 Justificativa

Sistemas são cada vez mais utilizados nas práticas de ensino, abrangendo desde a definição de aulas a ferramentas de auxilio no ensino. E cada vez mais é cobrado, não somente de alunos mais também de professores o domino sobre o computador.

A ciência da computação observa o mundo a sua volta e perceber problemas que podem e devem ser solucionados utilizando softwares, pois muitas vezes grandes

problemas podem ser facilmente resolvidos com a implantação de um sistema funcional e de fácil administração, até mesmo para alguém com pouca experiência no uso de computadores.

No caso especifico de gerar avaliações um sistema que faça isso de maneira automática, proporciona uma ferramenta para o professor, se ocupando do trabalho de separar e elaborar uma avaliação para determinada turma e se torna um aliado para os acadêmicos, garantindo que cada avaliação siga a mesma metodologia sem assim criar espaços para divergências no conteúdo cobrado e no formato de prova.

1.2 Objetivo do Trabalho

1.2.1 Objetivo Geral

Desenvolvimento de uma ferramenta para geração de avaliações com opções para definição de quantidade de questões, dificuldade e tipo. Facilitando a criação de teste pelos professores.

1.2.2 Objetivos Específicos

Os objetivos específicos consistem no desenvolvimento de um sistema geração de avaliações com seguintes funcionalidades:

a) Geração de avaliação em arquivo de formato PDF;

b) Geração de interface para edição de questões;

c) Criar uma opção para professores em relação à criação de testes.

II FUNDAMENTAÇÃO TEÓRICA

2.1 Desenvolvimento de Software

Com o aumento de uso de computadores a necessidade de se desenvolver programas para um grupo cada vez maior e mais exigente de usuários, o desenvolvimento de software se transformou de uma tarefa realizada por um só desenvolvedor, para um trabalho em equipe respeitando normas e padrões de projeto.

> O objetivo principal de um software é dar suporte aos usuários; Logo para o software ter sucesso, ele deverá atender as expectativas e necessidades dos usuários que utilizarão. Neste contexto, a palavra usuário não se refere apenas a pessoas, ela pode designar também outros sistemas que tenham de interagir com o software em questão. (MARTINA,2010, p. 314)

Manter o nível de desenvolvimento com uma equipe focada, respeitando padrões, é um desafio gigantesco para as empresas, pois desvios de prazo e pressões vindas da diretoria e clientes, fazem que em muitas vezes a qualidade de código seja esquecida para que o produto seja entregue na data combinada.

2.1.1 Modelo Cascata

O mais antigo simples dos modelos, resumidamente consiste em depois de definido os requisitos segue uma série de processos bem definidos que não podem ser modificados ou ignorados durante o desenvolvimento.

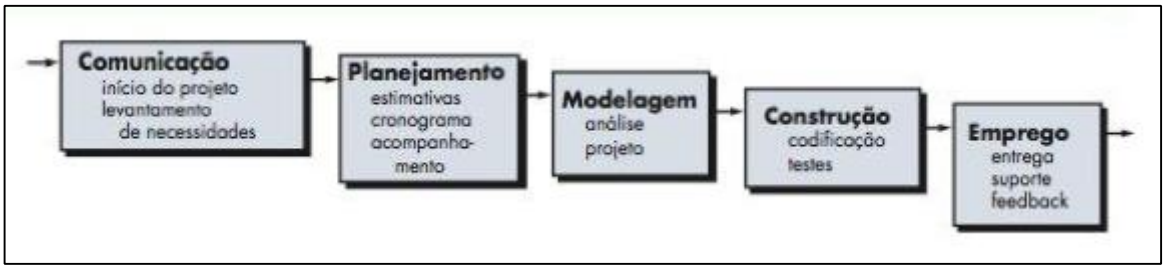

Figura 1–Modelo cascata
Fonte: PRESSMAN, 2011,p.60

Como visto na figura 1linearidade do modelo em cascata transforma o desenvolvimento em algo objetivo com metas definidas, porem essa arquitetura engessada muitas vezes pode esconder problemas em determinadas etapas atrasando os processos seguintes quando descobertos.

2.1.2 Prototipação

Como o modelo cascata (Figura 2) firma seus requisitos antes do desenvolvimento, o modelo de prototipação estabelece que depois de definido o requisitos iniciais seja criado um protótipo para validar a arquitetura.

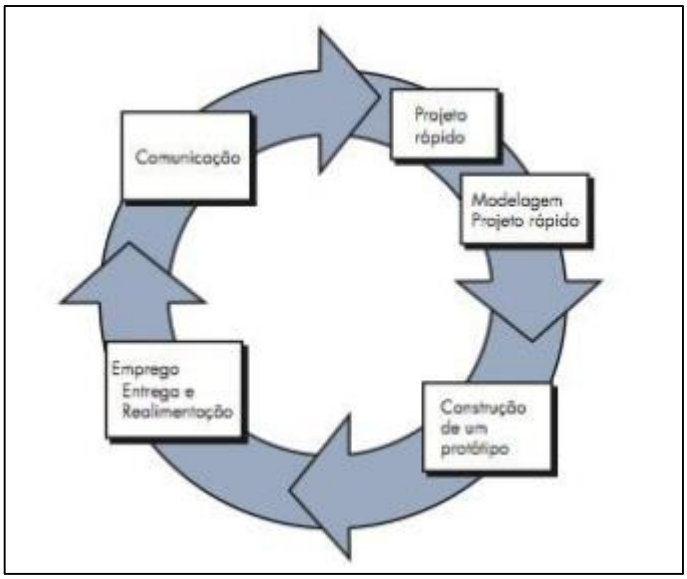

Figura2 -Modelo Prototipação
Fonte: PRESSMAN, 2011,p.63

O modelo de prototipação apresenta para o cliente uma ideia de como será o sistema, o que pode levar ao desinteresse por parte do cliente, devido à falta de uma interface amigável e problemas de código. Mas a experiência adquirida e o controle sobre a aplicação são muito maiores na fase de desenvolvimento, pois o desenvolvedor tem oportunidade de validar os requisitos e modificá-los se necessário for.

2.1.3 Desenvolvimento interativo

Unindo o princípio da prototipação com o modelo em cascata o desenvolvimento interativo propõe a construção do sistema de forma incremental. Invés de criar todo o sistema às interações são implementadas e uma a uma são incorporadas ao sistema (PRESSMAN, 2011,p.61). Sendo que, como o desenvolvimento não está fixado como no modelo em cascata, modificações pode ser feita durante ou após casa interação ser incorporada ao sistema.

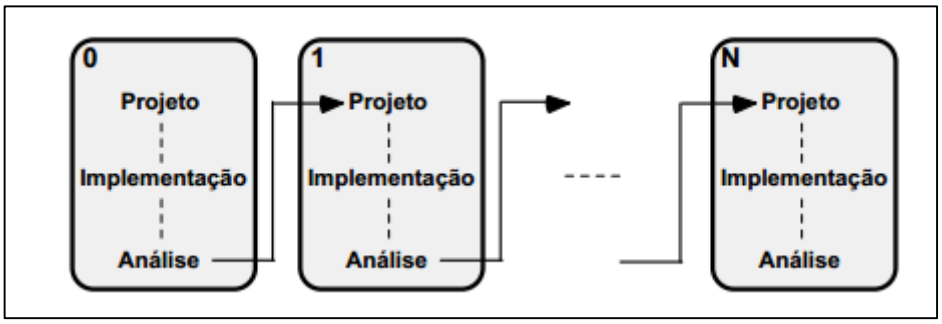

Figura 3-Desenvolvimento interativo
Fonte: < http://felipelirarocha.wordpress.com/2012/04/15/diversos-modelos-de-desenvolvimento-de-software-resumo/ >

Durante o processo de desenvolvimento as etapas do processo ficam registradas em uma lista de controle de projeto, que indica em qual etapa o desenvolvimento se encontra e quantas etapas faltam para a última. Cada etapa é dividida em: Projeto, implementação e analise como representado na figura 3. Se segue esse controle, até que a lista esteja vazia.

2.1.4 Modelo Espiral

O modelo, como sugere o nome, é composto de diversos ciclos. “Um modelo espiral é divido em um conjunto de atividades metodológicas definidas pela equipe de engenharia de software” (PRESSMAN, 2011,p.63). Cada ciclo contém o que é necessário para a resolução do objetivo, desde sua edificação e os possíveis caminhos para conclui-lo.

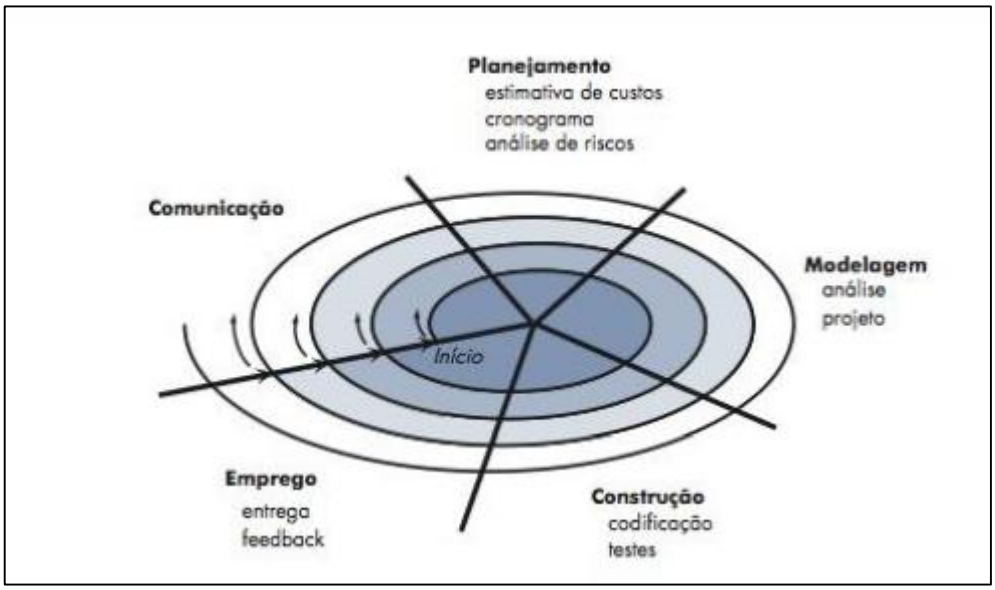

Figura 4-Modelo Espiral
Fonte: PRESSMAN, 2011,p.63

A evolução do projeto dentro do modelo espiral se dá pela continuidade, cada etapa do ciclo elimina ou cria novos riscos que são levados em consideração na próxima etapa, pois cada ciclo termina com uma revisão avaliando o produto até o momento e criando o plano para o próximo passo.

2.1.5 Técnicas de quarta geração 4-GT

As técnicas de quarta geração ou linguagens de 4° geração (4GL's) consiste em linguagens de alto nível usadas para aplicações especificas. Onde a velocidade de implementação pode ser maximizada pelo uso de ferramentas.

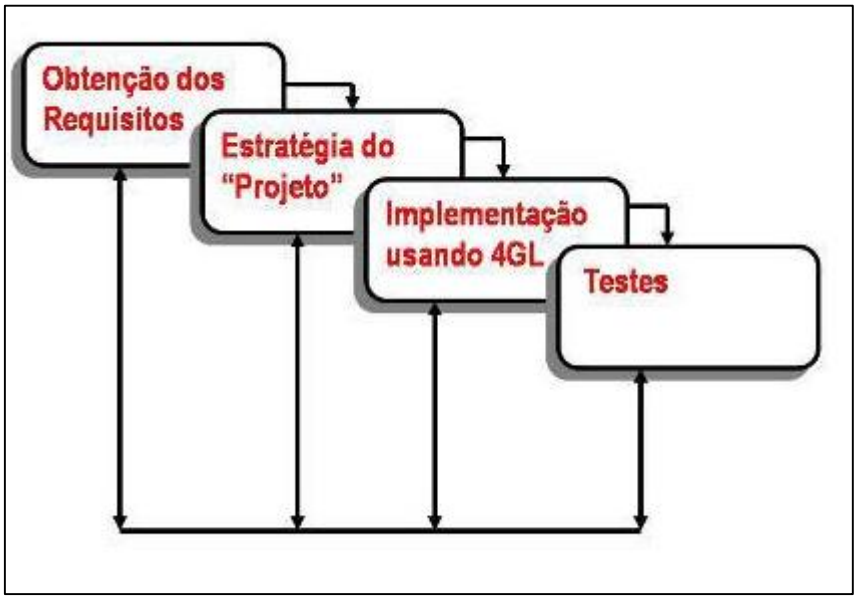

Figura 5 -4-GL's
Fonte< http://centraldaengenharia.wordpress.com/2011/02/16/paradigmas-4geracao/>

Ainda que seja uma forma de ação rápida os mesmos cuidados devem ser tomados para não levar aos mesmos problemas de outros modelos, deve ser tomar cuidado na qualidade, manutenção e aceitação pelo cliente.

2.2 C#

Segundo Watson (2010, p.378) "C# é uma linguagem simples, poderosa, com tipagem segura e orientada a objetos". Compatível com o .NET Framework e implementada através da IDE Visual Studio, da Microsoft, pode ser utilizada para criação

de aplicações Desktop para Windows e sistemas WEB.Com uma sintaxe similar ao c++ e Java, desenvolvedores dessas linguagens tem grande facilidade em compreender e produzir códigos utilizando o c#. Sendo mais Simples que o c++ e disponibilizando de mais recursos como uso de delegações expressão lambda e privilégios ao acesso a memória não presentes no Java.

> O processo de compilação do C# é simples comparado com C++ e mais flexível que em Java. Não há arquivos de cabeçalho separados, e não há a necessidade de que métodos e tipos sejam declarados em uma ordem específica. Um arquivo de código em C# pode definir qualquer número de classes, estruturas, interfaces e eventos. (MICROSOFT corp., 2014).

Como é orientada a objetos o C# faz uso dos conceitos de Herança Polimorfismo e encapsulamento. Todas as suas definições de métodos e variáveis são obrigatoriamente feitas em definições de classes. Uma classe pode herdar apenas de uma classe, mas pode implementar interfaces sem restrições de quantidade. Como é uma linguagem criada e usada para uso no ambiente Windows acesso a objetos COM e DLLs nativas do sistema o C# pode fazer praticamente tudo como uma aplicação c++ faria.

2.2.1 NET Framework

Códigos criados utilizando C# são executados utilizando o .NET Framework (versão 4.5 lançada em 2012) um componente Windows que faz uso de princípios utilizados na tecnologia Java, os programas são compilado duas vezes uma na distribuição e outra na execução. O Framework inclui um sistema de execução virtual (CLR-*Commom Language Runtime*)."O CLR é a implementação comercial da Microsoft da infraestrutura de linguagem comum (CLI), um padrão internacional que é a base para a criação e execução de ambientes de desenvolvimento em que as linguagens e as bibliotecas trabalham juntos sem problemas”.

> O código-fonte escrito em C# é compilado em uma linguagem intermediária (IL) que está em conformidade com a especificação CLI. O código IL e recursos, como bitmaps e strings, são armazenados no disco em um arquivo executável chamado de um assembly, normalmente com uma extensão .exe ou .dll. Um assembly contém um manifesto que fornece informações sobre os tipos do assembly, versão, cultura e requisitos de segurança. (MICROSOFT corp., 2014)

Depois que é executado, o programa c#, o *aseembly* é carregado para a execução virtual (CLR- *Commom LanguageRuntime*) que define as ações a serem tomadas com base nas informações do manifesto.Com os pré-requisitos aprovados o CLR compila para Just in time(JIT) convertendo a linguagem intermediaria em instruções de máquina.

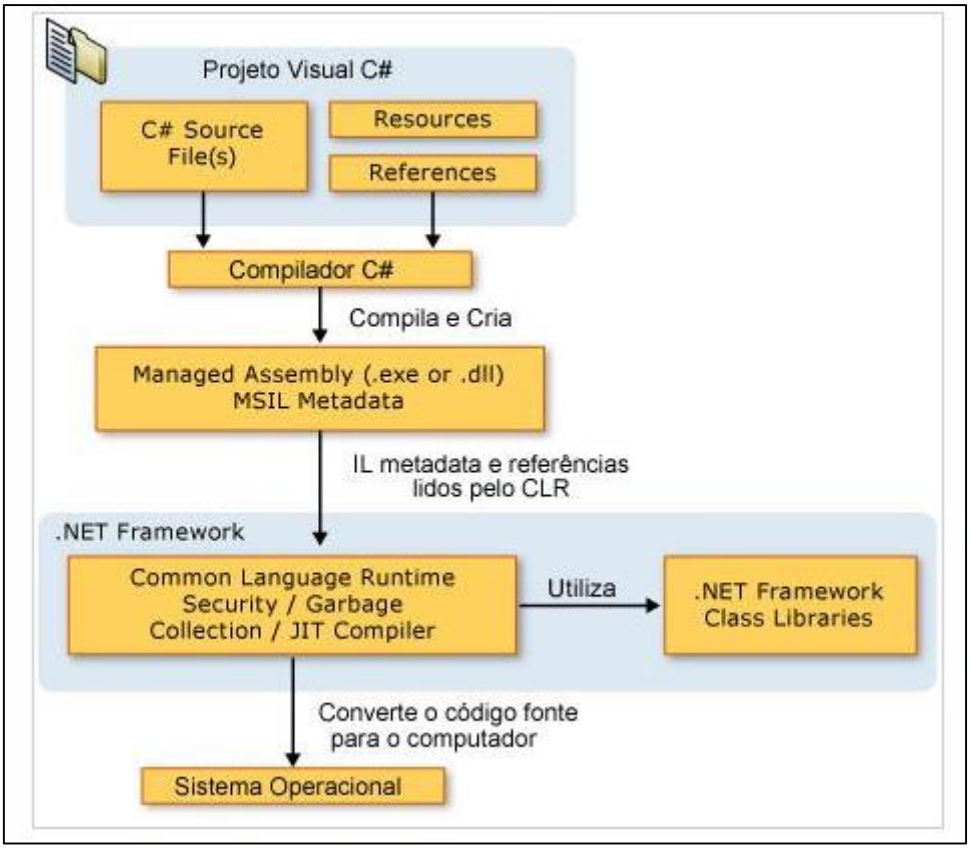

Figura 6- Fluxo de compilação
Fonte<http://msdn.microsoft.com/pt-br/library/z1zx9t92.aspx>

O código intermediário gerado pelo compilador c# (figura 6) fica de acordo com as especificações de tipo comum (CTS- *Common Type Specification*) podendo assim ser referenciado ou referenciar código de versões antigas do framework ou em qualquer linguagem compatível com CTS.

2.3 Banco de Dados

Segundo Date (2003, p 3) "Um Sistema de banco de dados é basicamente apenas um sistema computadorizado de manutenção de registros". Tal como há várias maneiras de se guardar registros fisicamente, os bancos de dados utilizam diferentes formas de armazenar e descrever as informações registradas.

2.3.1 SQL (Structured Query Language)

A Linguagem de Consulta Estruturada (em inglês: *Structured Query Language*) é uma linguagem para consulta e manipulação para bancos de dados relacional, criada no começo dos anos 70 pelos laboratórios da IBM.

> IBM, junto com outros fornecedores de banco de dados relacionais, queria um método padronizado para acessar e manipular dados em um banco de dados relacional. Embora IBM tenha sido a primeira a desenvolver a teoria de banco de dados relacional, a Oracle foi a primeira a comercializar a tecnologia. (KLINE,2010)

Desde o princípio das bases de dados relacionais, diversas linguagens fazem acesso e manipulam dados, mas nenhuma é tão fácil de ser aprendida e utilizada como é o SQL, assim hoje o SQL é o padrão para acesso a bancos de dados relacionais.

2.3.2 Microsoft SQL Server CE

O SQL Server CE pode ser definido como: "...é um banco de dados relacional compacto produzido pela Microsoft para aplicativos que são executados em dispositivos móveis e desktops. Antes da introdução da plataforma de desktop, ele era conhecido como SQL Server para Windows CE e SQL Server Mobile Edition. " (MICROSOFT, 2014). Utiliza o T-SQL (*Transact*-SQL) para manipulação de dados, o T-SQL é uma variação da Microsoft do SQL criado pela IBM, facilitando o a comunicação das aplicações, pois todas as aplicações que se comunicam com o SQL server CE necessitam usar o T-SQL.

2.4 Programação Orientada a Objetos

O paradigma de programação orientada a objetos é o mais utilizado por grandes corporações na hora de desenvolver sistemas, pois possui um conjunto de ideias, conceitos e abstrações muito similar ao mundo real. Segundo (LASSALA, 2014) "...principais vantagens da POO são o reuso de código e a capacidade de expansão do mesmo. "

A linguagem orientada a objetos tem como características principais: Abstração, Encapsulamento, Herança e Polimorfismo.

"Abstração pode ser definida como a capacidade de representar cenários complexos usando termos simples" (LASSALA, 2014). Em programação é utilizado esse conceito para simplificar objetos complexos facilitando o uso pelo desenvolvedor. Esses objetos podem tanto ser virtuais como um campo de texto em uma "janela" do sistema quantos objetos complexos reais um carro ou um produto por exemplo.

2.5 Padrões de projeto

Padrões de projeto são o guia para o desenvolvimento de qualidade, ajudam o desenvolvedor a achar o caminho certo mais rápido e permitem que o novo desenvolvedor compreenda o sistema com mais facilidade. Segundo Gamma at al um padrão de projeto tem quatro elementos: o nome do padrão que consiste em explicar o problema a solução e as consequências em um ou duas palavras, o problema que revela o contexto da situação, a solução que descreve a ação tomada as relações e responsabilidades do padrão e as consequências mostrando o resultado da utilização do padrão.

> Padrões de projeto não são sobre desenhos, como listas ligadas e tabelas de hash que podem ser codificados em classes e reutilizados como tal. Nem são, projetos específicos de domínio complexos para um aplicativo inteiro ou subsistema. (GAMMAat al,2002, p 20)

Gamma at al os padroes de projeto podem ser divididos segundo sua finalidade, cada padrao pode atuar em uma parte diferente da criação de um software.Segundo essa definição a divisao dos padroes são : Criação, estruturais e comportamentais.

2.5.1 Padrões de Criação

Os padrões de criação têm como objetivo controlar e abstrair os processos de criação de um objeto, agindo na iniciação de um objeto o sistema não fica preso a criação do mesmo, protegendo a continuidade do sistema caso seja alterada a definição para criação desse objeto. Segundo Gamma at al os padrões de criação são: *Factory method, Abstrac Factory, Builder, Prototype e Singleton.*

2.5.1.1 Singleton

Singleton (coisa única, em tradução livre) define a criação de um único objeto para determinada classe, após ser iniciado o sistema deve garantir que outros objetos do mesmo tipo não sejam criados e utilizados.

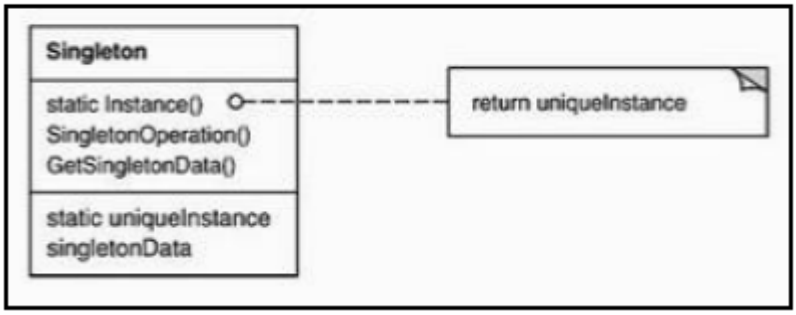

Figura 7 – Estrutura do Singleton
Fonte: GAMMA et al, 2000, p.130

Como mostra a figura 7, é sempre retornado a mesma instancia do objeto desejado pode ser utilizado por exemplo, em um sistema que necessite fazer uma conexão a uma base de dados, um objeto *singleton* não permitiria a criação de várias conexões desnecessárias que tornariam o sistema lento.

2.5.1.2 Factory Method

O padrão *Factory Method* define a criação de objetos deixando que o sistema em tempo de execução decida qual classe deve ser instanciada.

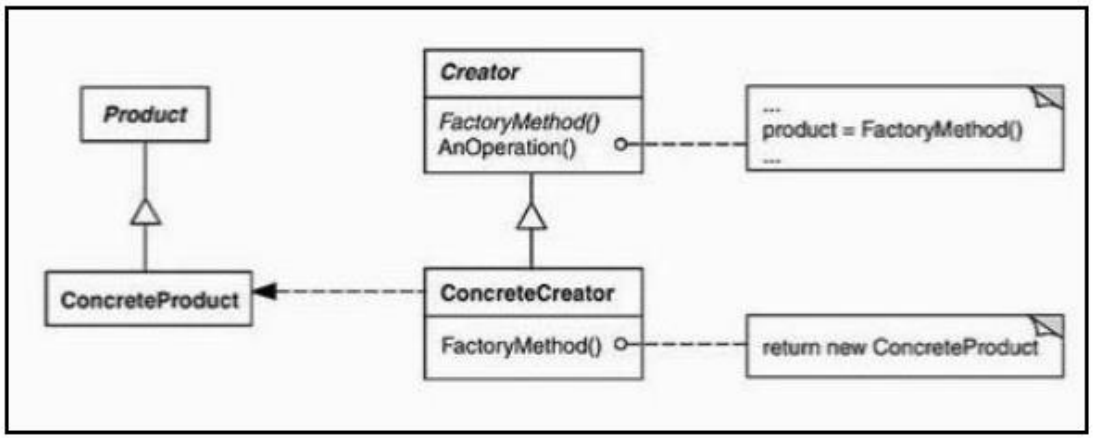

Figura 8 – Estrutura do Factory Method
Fonte: GAMMA et al, 2000, p.113

Na figura 8 pode se observar as 4 estruturas básicas do padrão:*Creator* que declara o *factory method*,*ConcreteCreator* responsável por sobrescrever o *factory method* e retornar um objeto da classe*ConcreteProduct*,O *Product* define uma interface para os objetos e o *ConcreteProduct*define a implementação para a interface.

2.5.1.3Abstract Factory

Como no padrão *Factory Method*a criação de objetos ocorre em tempo de execução, entretanto na *Abstract Factory*(figura 9) a criação utiliza um objeto conhecido.

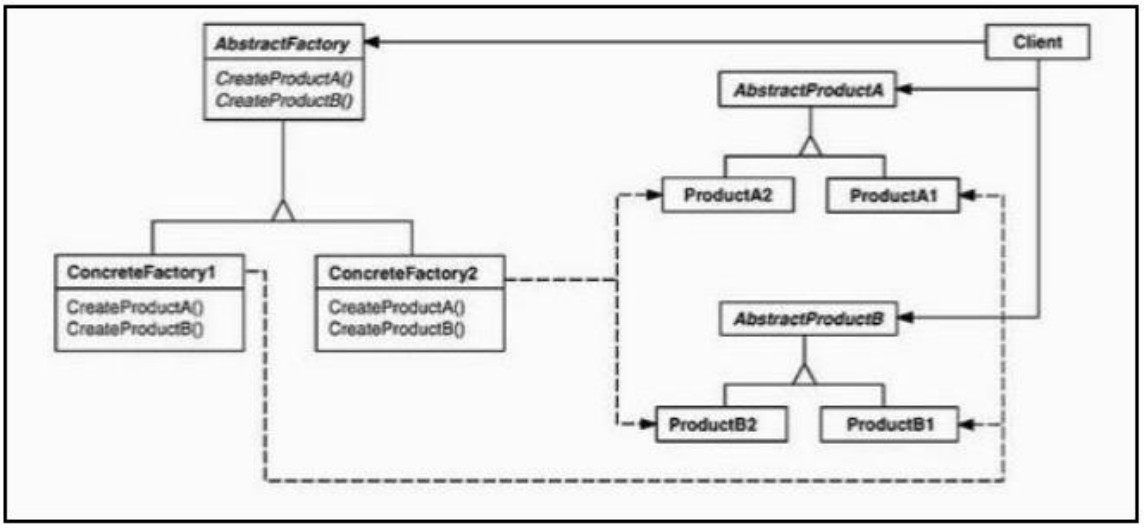

Figura 9– Estrutura do Abstract Factory
Fonte: GAMMA et al, 2000, p.97

Assim um objeto cliente utiliza apenas objetos criados pela classe abstrata sem precisar ter conhecimento da classe concreta, porem cria a dificuldade de implementar novos tipos.

2.5.1.4 Prototype

Define a criação de um objeto protótipo que quando solicitado por um cliente retorne uma cópia de si mesmo. Permitindo a adição ou remoção de objetos em tempo de execução.

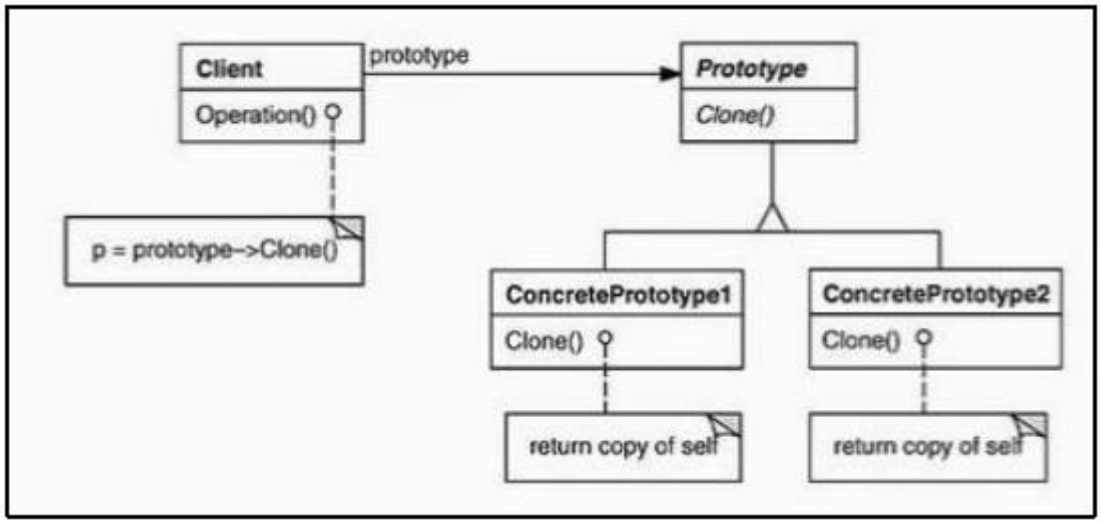

Figura Figura 10 – Estrutura do Prototype
Fonte: GAMMA et al, 2000, p.123

Na figura 10 fica clara a forma como o padrão é implementado com: a classe *prototype* com a capacidade de clonar a si mesmo, um *ConcretePrototype* que é a implementação do *prototype* e pôr fim a classe *Client* que cria novos objetos através do *Prototype.*

2.5.1.5 Builder

Com o uso do padrão *Builder* (Figura 11) permite a separação da construção de um objeto complexo da sua representação, libertando o processo de construção para criar diferentes representações.

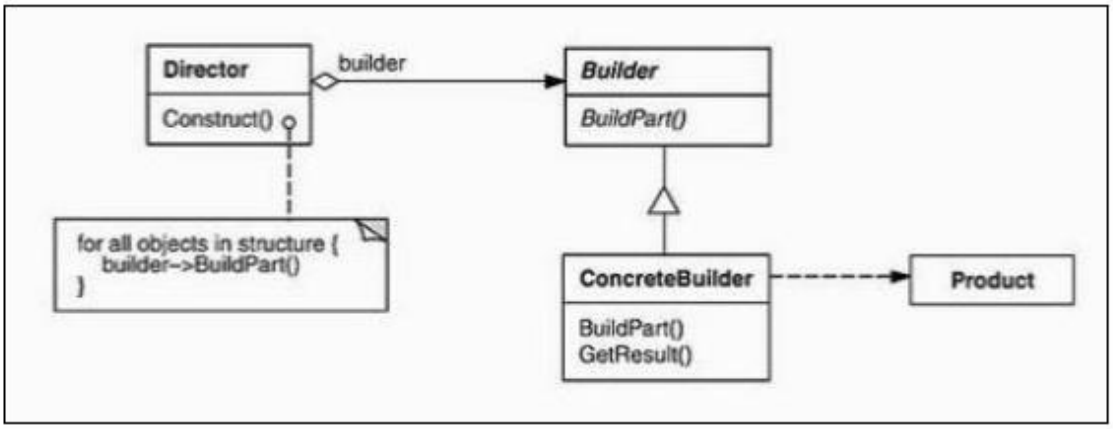

Figura 11 – Estrutura do Builder
Fonte: GAMMA et al, 2000, p.105

Na estrutura *Builder* o *Director*é responsável por construir um objeto, o Builderpor especificar uma interface para o construtor, ConcreteBuilder mantém a representação para recuperação do produto e o Product que é o objeto final.

2.5.2 Padrões Estruturais

O padrão estrutural como o próprio nome revela, se preocupam com a estrutura do sistema de como as classes e objetos vão ser compostos. Agindo diretamente no design do sistema encontrando maneiras de efetuar o relacionamento entre as entidades. *Adapter, Bridge, Composite, Decorator, Facade, Flyweight e Proxy* são os padrões estruturais segundo Gamma at all.

2.5.2.1 Proxy

No padrão *Proxy* (Figura 12) o sistema utiliza um objeto substituto até que objeto real seja carregado ou outro objeto seja inicializado no sistema.

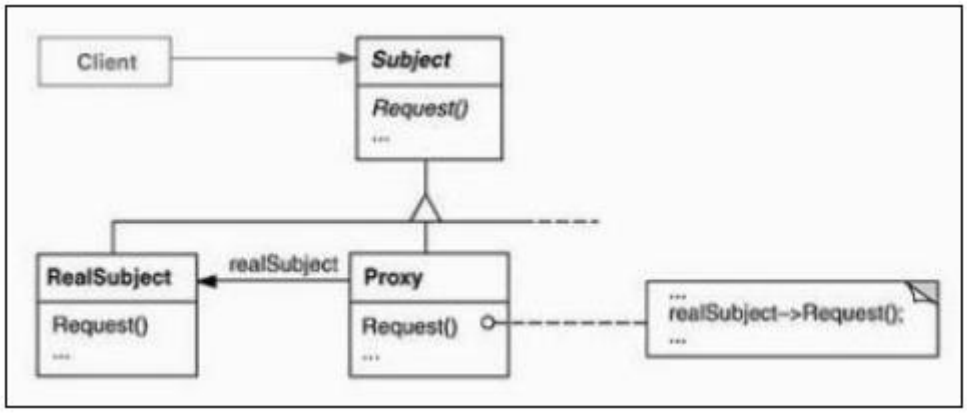

Figura 12 – Estrutura do Proxy
Fonte: GAMMA et al, 2000, p. 200

Pode ser usado por exemplo em um sistema que demora para carregar um grande arquivo, o objeto *proxy* teria a função de mostrar o quanto desse objeto já foi carregado para o usuário, assim que o processo termina o objeto *proxy* pode ser descartado.

2.5.2.2 Adapter

O padrão tem como função adaptar um objeto a uma nova interface. Sendo que para classes (figura 14) utiliza herança múltipla e para objetos (figura 13), composição.

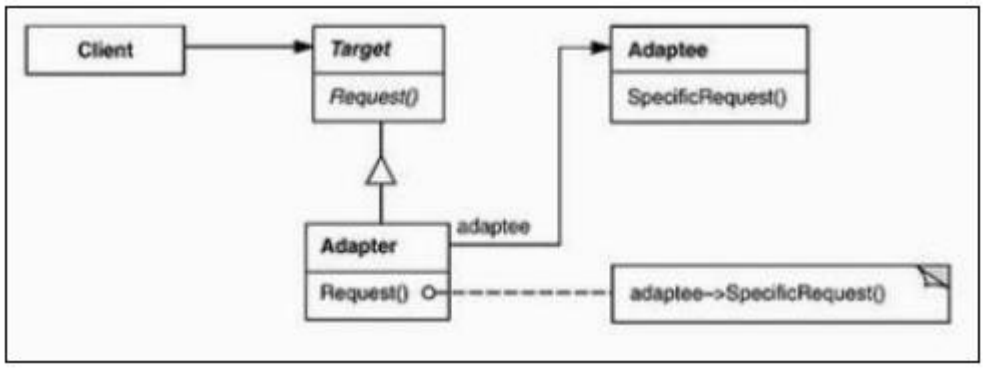

Figura 13– Estrutura do Adapter nível de objeto
Fonte: GAMMA et al, 2000, p. 142

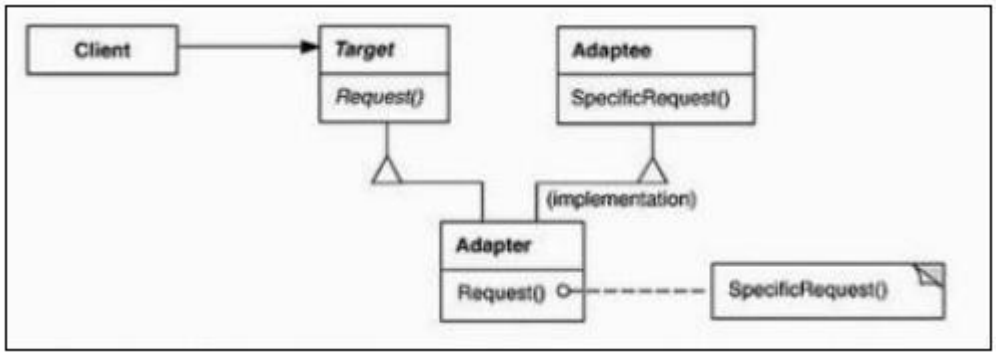

Figura 14– Estrutura do Adapter nível de classe
Fonte: GAMMA et al, 2000, p. 142

Também chamado de *Wrapper* permite que classes com interfaces diferentes possam interagir. Fazendo que o cliente utilize outros objetos a partir de uma interface única.

2.5.2.3 Bridge

Utiliza-se o padrão *Bridge* (Figura 15) quando é necessário que uma interface varie independentemente do seu uso no sistema.

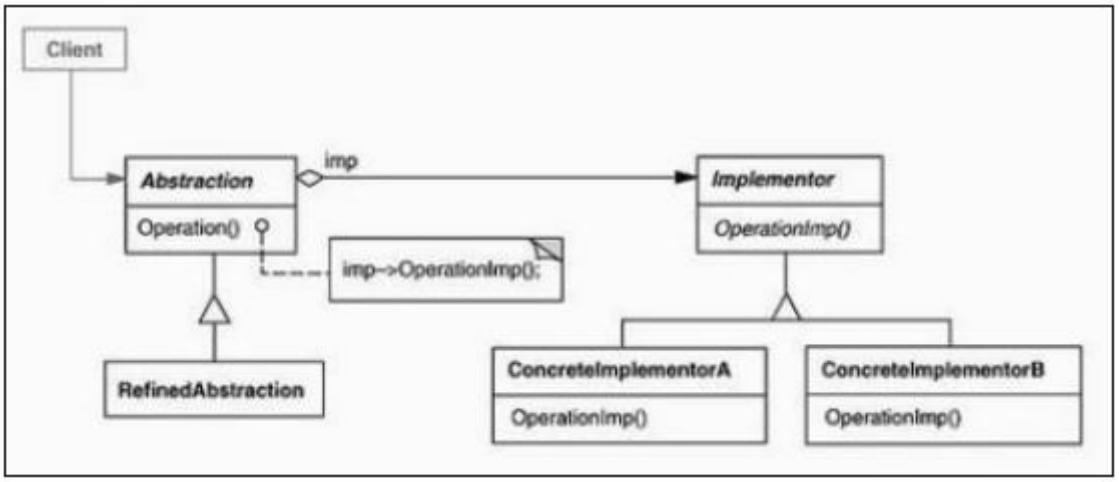

Figura 15 – Estrutura do Bridge
Fonte: GAMMA et al, 2000, p. 153

Assim o padrão remove a abstração do objeto para que ambos, objeto e interface, possam agir de forma distinta.

2.5.2.4 Composite

Implica que cada elemento de uma estrutura comum implementem a mesma interface ou classe, tratando de forma uniforme objetos individuais.

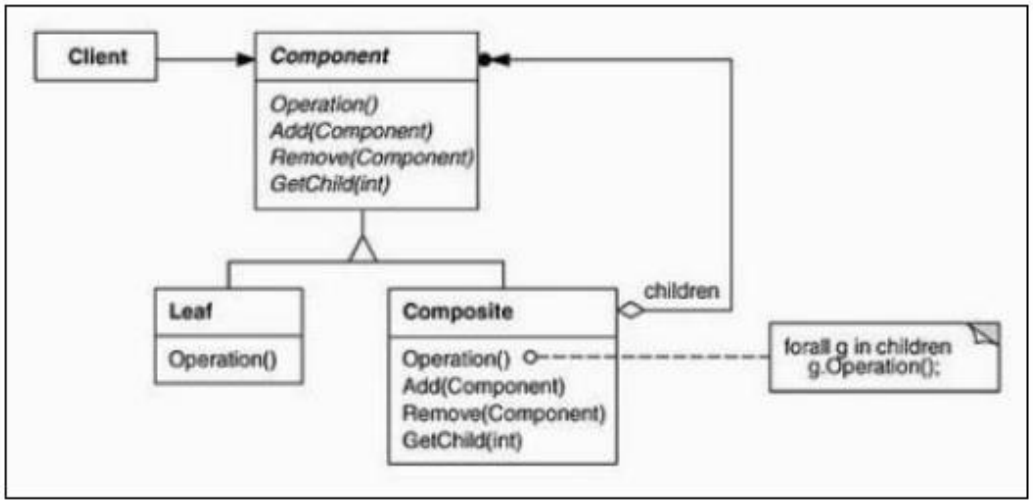

Figura 16 – Estrutura do Composite
Fonte: GAMMA et al, 2000, p. 161

Como pode ser observado na figura 16 todos os métodos comuns podem ser aplicados aos objetos compostos.

2.5.2.5 Facade

O padrão *Facade* (Figura 17) Consiste na criação de uma interface simplificada para utilização de sub-rotinas. A interface serve como fachada para esconder a complexidade dos métodos a serem utilizados.

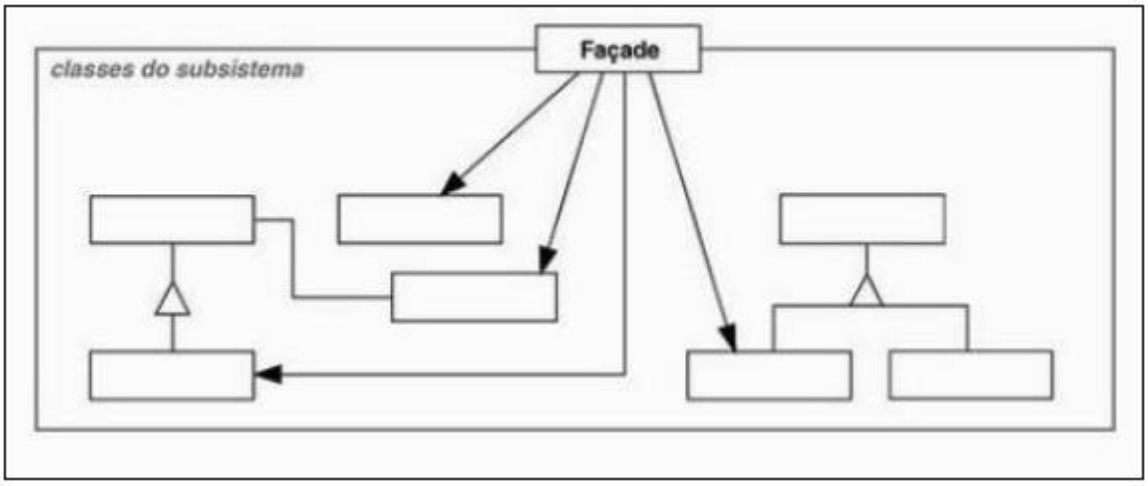

Figura 17 – Estrutura do Facade
Fonte: GAMMA et al, 2000, p. 181

Esse padrão facilita a implementação pelo cliente que não necessariamente precisa saber como as sub-rotinas efetuam seu trabalho, apenas utiliza o objeto fachada para realizar sua tarefa.

2.5.2.6 Decorator

O padrão *Decorator* (Figura 18) permite adicionar responsabilidades em tempo de execução a um objeto através de um decorador.

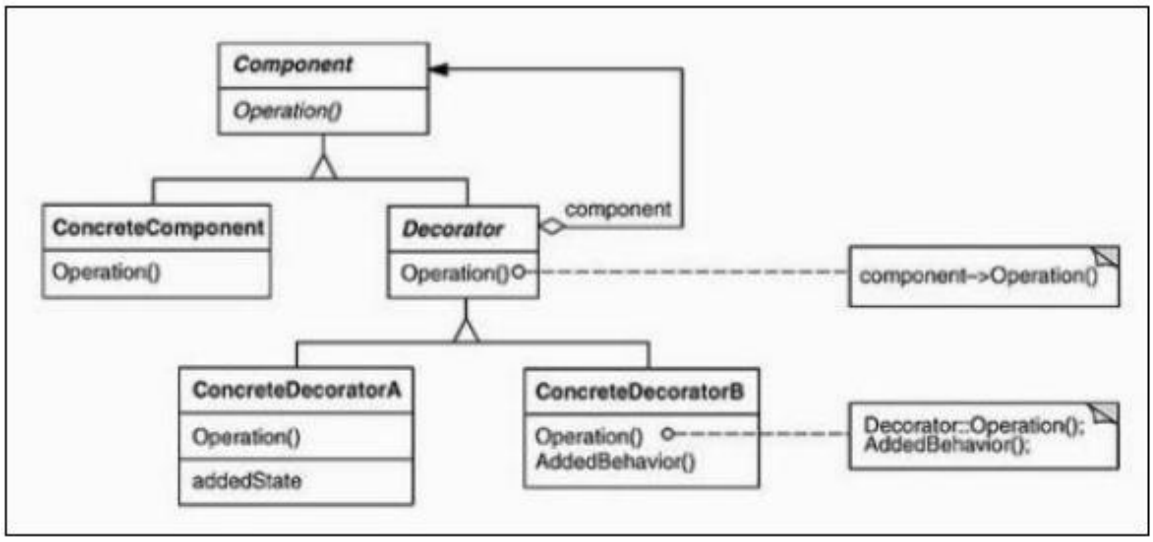

Figura 18 – Estrutura do Decorator
Fonte: GAMMA et al, 2000, p. 172

Apesar de não possuir as mesmas complexidades que a herança, torna comparações mais difíceis de serem efetuadas e se utilizado em demasia resulta em um design com muitos pequenos objetos similares.

2.5.2.7 Flyweight

O *Flyweight* (Figura 19) é utilizado quando é necessário manipular uma grande quantidade de objetos de forma eficiente.

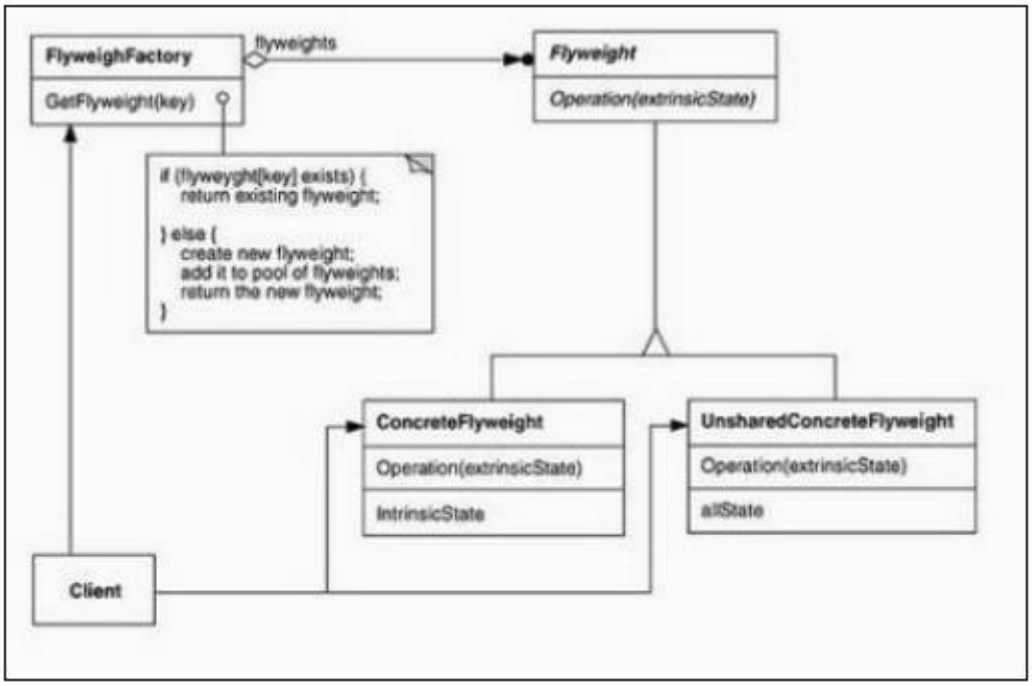

Figura 19 – Estrutura do Flyweight
Fonte: GAMMA et al, 2000, p. 190

Assim cada objeto contém referência para outro objeto, reduzindo a quantidade de memória necessária para acessar um determinado recurso.

2.5.3 Padrões Comportamentais

Esses padrões atuam na definição das responsabilidades que são atribuídas as entidades, facilitando a comunicação interna entre os objetos. Gamma at al define os seguintes padrões de comportamentais: *Interpreter, Template method, Chain of responsiblity, Command, Interator, Mediator, Memento, Observer, State, Strategy e Visitor.*

2.5.3.1 Memento

O padrão *Memento* (Figura 20) possibilita a um objeto capturar e salvar seu estado, para quando ou se necessário for, restaurar o objeto a condição original.

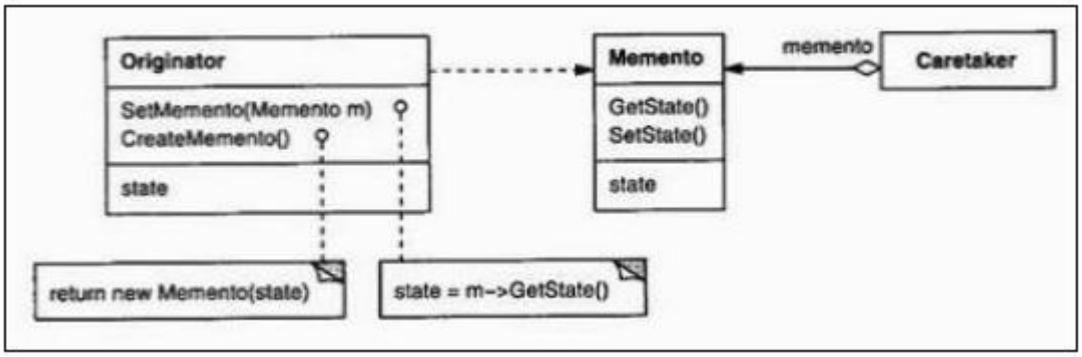

Figura 20 – Estrutura do Memento
Fonte: GAMMA et al, 2000, p. 268

Apresenta um custo considerável, pois utiliza três objetos para realizar a restauração, o objeto original, o objeto memento com as informações do estado original e o objeto "zelador" que contém as referências de associação dos objetos memento aos originais.

2.5.3.2State

Faz-se necessário utilizar o padrão *State* (Figura 21) quando é preciso modificar o comportamento de um objeto em relação ao seu estado atual.

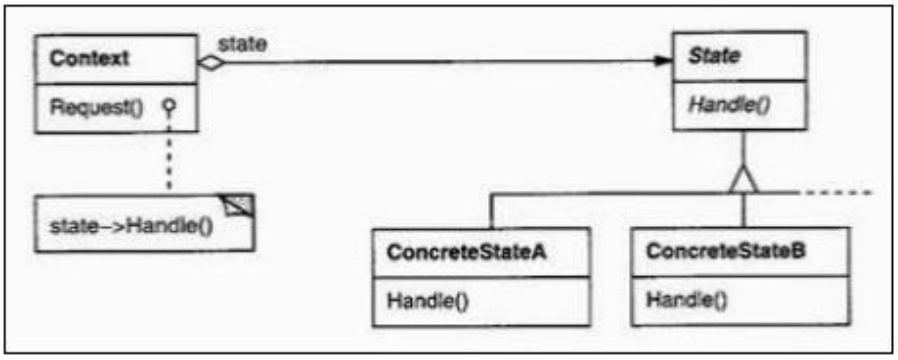

Figura 21 – Estrutura do State
Fonte: GAMMA et al, 2000, p. 285

Pode gerar problemas para gerenciar muitos comportamentos pois cria um objeto para cada estado, podendo ocasionar inconsistência de informação com um número elevado de estados.

2.5.3.3 Chain of Responsibility

O padrão *Chain of Responsibility* (Figura 22) permite que em tempo de execução qual objeto tratara uma mensagem, desacoplando remetente e receptor.

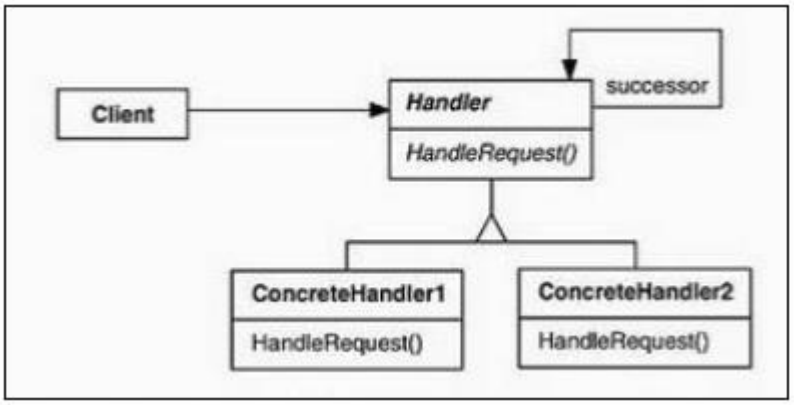

Figura 22 – Estrutura do Chain of Responsibility
Fonte: GAMMA et al, 2000, p. 214

Permitindo assim que cada objeto possa tratar uma mensagem ou passa-la para o próximo objeto na sequência.

2.5.3.4 Observer

Quando se tem um fraco acoplamento entre objetos, se utiliza o padrão *Observer* (Figura 23) para notificar os objetos dependentes de uma modificação no estado do objeto original.

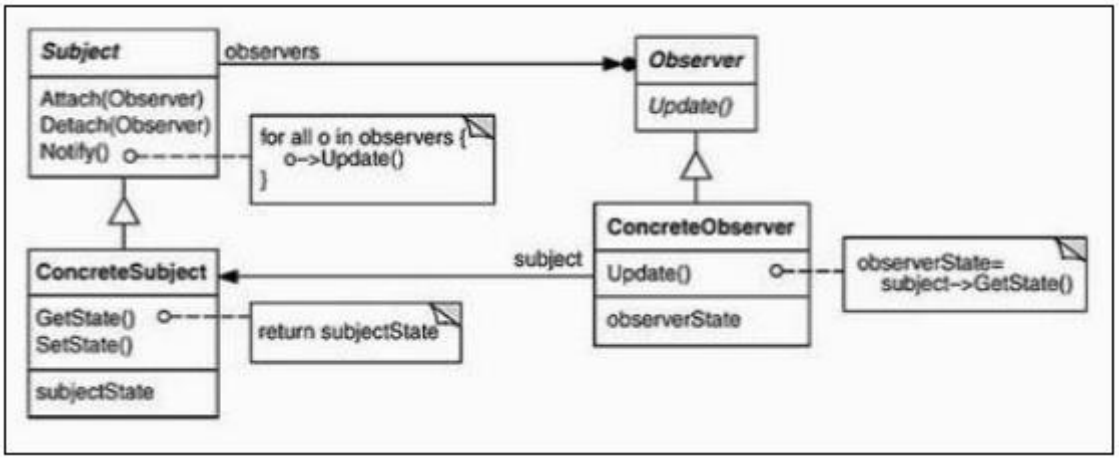

Figura 23 – Estrutura do Observer
Fonte: GAMMA et al, 2000, p. 275

Assim o objeto *Observer* deve saber os objetos de interesse, os *subjects*, que por sua vez são os responsáveis por notificar os observadores quando for modificado.

2.5.3.5 Strategy

É semelhante ao padrão *State* (Figura 21) mas ao invés de encapsular a informação do estado o *estrategy*(figura 24) encapsula um algoritmo.

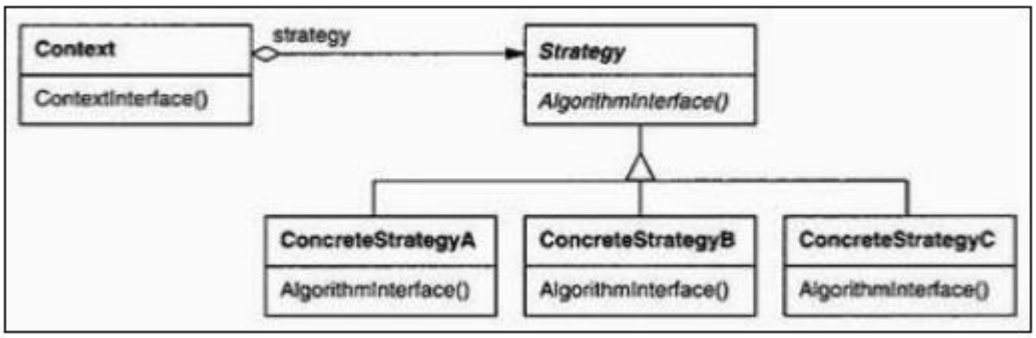

Figura 24 – Estrutura do Strategy
Fonte: GAMMA et al, 2000, p. 294

Com esse padrão pode-se variar a resolução de um problema para cada cliente que o utiliza o objeto dependendo da necessidade.

2.5.3.6 Template Method

O padrão *Template method* (Figura 25) define que cada objeto implemente um algoritmo definido por uma superclasse.

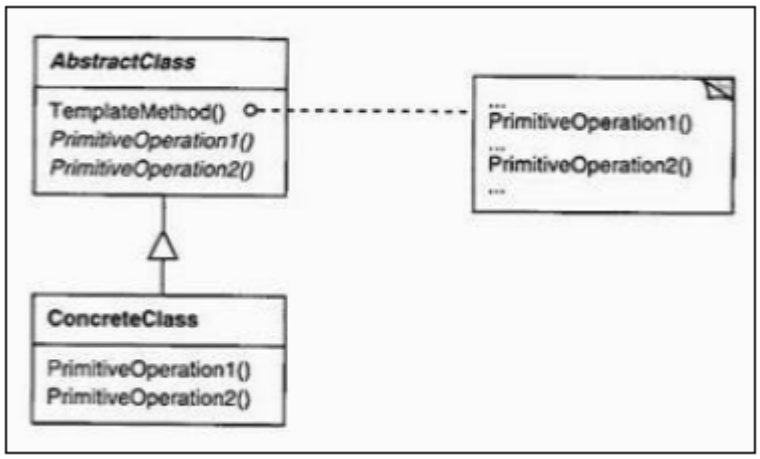

Figura 25 – Estrutura do Template Method
Fonte: GAMMA et al, 2000, p. 302

O padrão permite alterar parte desse algoritmo herdado caso seja necessário sem mudar a estrutura original do mesmo.

2.5.3.7 Command

O *command* (Figura 26) define que um comando ou instrução possa ser encapsulado, para que possa ser utilizada em entre diversos objetos.

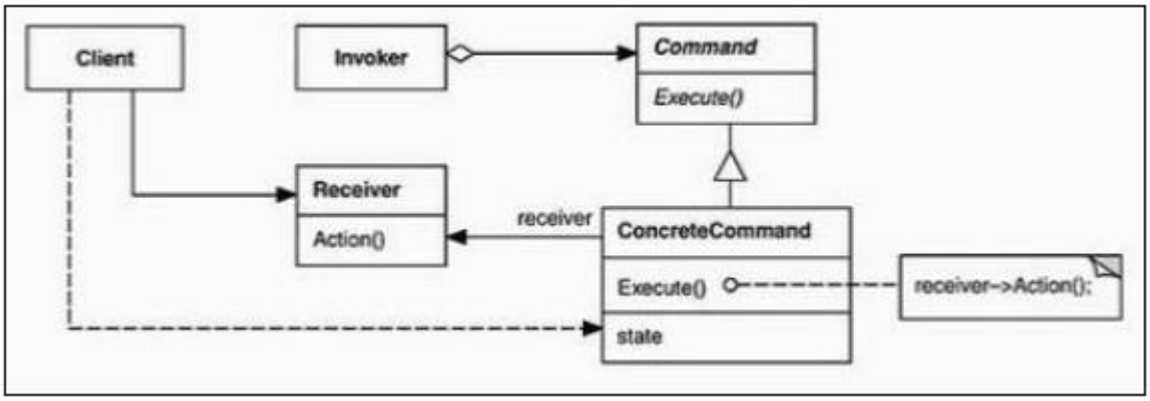

Figura 26 – Estrutura do Command
Fonte: GAMMA et al, 2000, p. 225

Facilita a parametrização de objetos por ações a serem executadas e também suporta operações que possam ser desfeitas.

2.5.3.8 Iterator

O padrão *Interator* (Figura 27) permite que os objetos acessem diferentes estruturas de dados da mesma forma sem precisar conhecer o comportamento individual das estruturas.

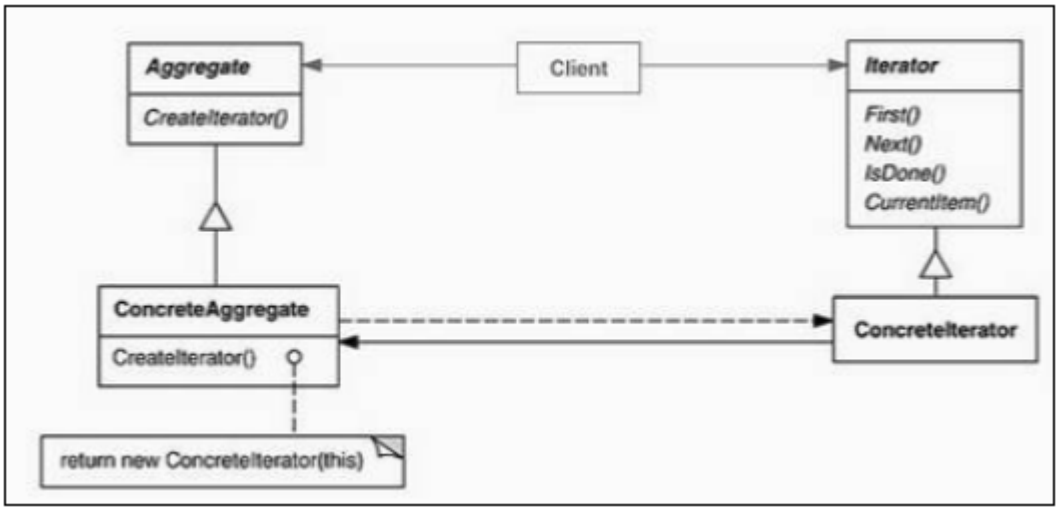

Figura 27 – Estrutura do Iterator
Fonte: GAMMA et al, 2000, p. 246

O padrão simplifica a interface do objeto e cada estrutura programa os métodos comuns de uma interface para acesso de seus dados.

2.5.3.9 Interpreter

O padrão *Interpreter* (Figura 28) tem como objetivo a capacidade de estender compreender sentenças de uma determinada linguagem.

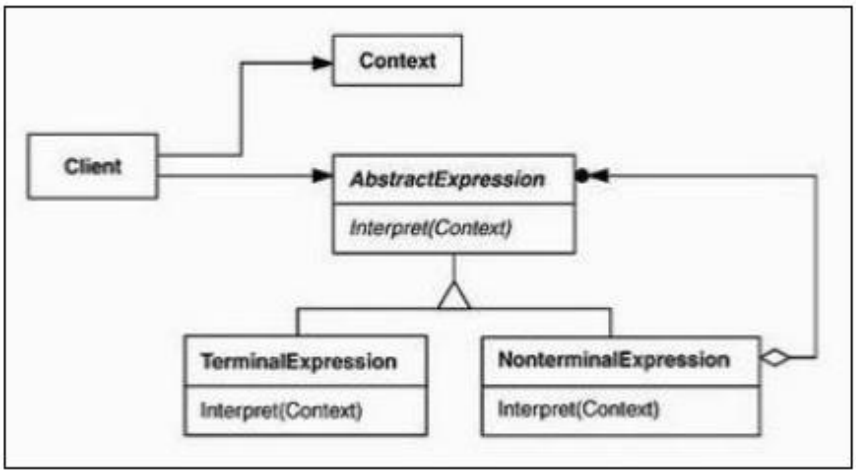

Figura 28– Estrutura do Interpreter
Fonte: GAMMA et al, 2000, p. 234

Assim o padrão busca que com o uso de uma linguagem formal simples, resolver problemas recorrentes.

2.5.3.10 Mediador

O padrão *Mediato* (Figura 29) a comunicação entre os objetos é centralizada e encapsulada, reduzindo a dependência e limitando o uso de subclasses.

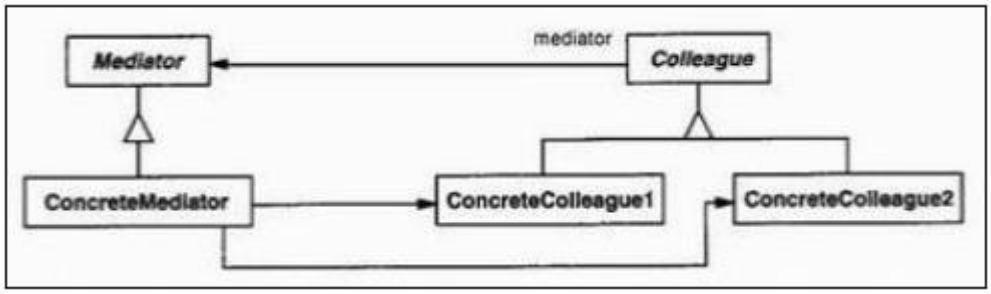

Figura 29 – Estrutura do Mediator
Fonte: GAMMA et al, 2000, p. 260

A estrutura do *Mediator* se define*: Mediator*a interface para a comunicação entre objetos, *ConcreteMediator* a interface que coordena a comunicação e o *ConcreteColleague*que se comunicam com outros objetos através do *mediator*.

2.5.3.11 Visitor

O *Visitor* (Figura 30) permite criar uma nova operação sem a necessidade de mudar a classe dos elementos sobre as quais ela opera.

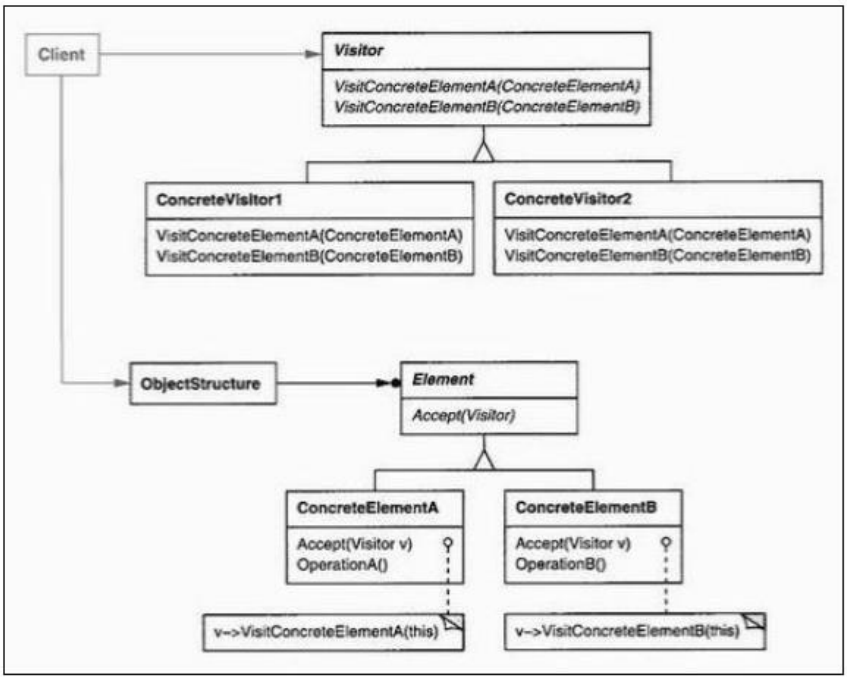

Figura 30 – Estrutura do Visitor
Fonte: GAMMA et al, 2000, p. 308

É uma maneira de separar os algoritmos de um objeto. Adiciona novas funcionalidades a estruturas de um objeto já existente no sistema sem a necessidade de modificá-lo.

III METODOLOGIA

3.1 Tipo de Pesquisa

Para o presente trabalho foi utilizado a pesquisa exploratória definhada por Theodorson e Theodorson (1970) "Estudo exploratório. Um estudo preliminar o principal objetivo é familiarizar-se com um fenômeno que é o de investigar, de modo que o maior estudo a seguir pode ser projetado com uma maior compreensão e precisão". E fazendo o uso de revisão bibliográfica sobre os diversos pontos explorados para a resolução do objetivo.

3.2 Método Utilizado

Para elaboração desse trabalho foi utilizado o método de Pesquisa bibliográfica, foram utilizados livros, *Websites*, periódicos para definir os conceitos de padrões de projeto e sua utilização no desenvolvimento de software.

Realizou-se um estudo sobre programação orientada a objetos e persistência em base de dados, e também uma análise sobre os padrões de questões aplicadas em provas de nível nacional, como a PosComp organizada pela Sociedade Brasileira de Computação e o Exame Nacional de Desempenho de Estudantes organizado pelo ministério da educação.

IV.PROJETO

Para o desenvolvimento de qualquer sistema computacional é preciso identificar as necessidades e o perfil do usuário final, adaptando assim suas funcionalidades, requisitos e o hardware necessários para a completa utilização do sistema.

Para criação do TCS (*Test Creator System*), foi utilizada a linguagem C# através do *Visual Estudio*, que se torna perfeito para criação de sistemas *desktop* com é o caso do TSC.

O princípio básico do sistema é a criação de provas automaticamente, com uma interface simples e menus intuitivos o sistema é de fácil utilização independentemente da intimidade com o uso do computador do usuário. É um sistema de instalação simplificado e disponibilidade para *download* na nova página da instituição.

Se difere de outros sistemas semelhante pois cria uma avaliação com matéria e disciplina determinadas, enquanto outros sistemas que utilizam um sistema de perguntas e respostas apenas.

4.1 Diagramas de Classes

Diagrama de classe do modelo do sistema (Figura 31). Representa a estrutura base do sistema, tanto interface quanto funcionalidades são criadas com integração com o modelo.

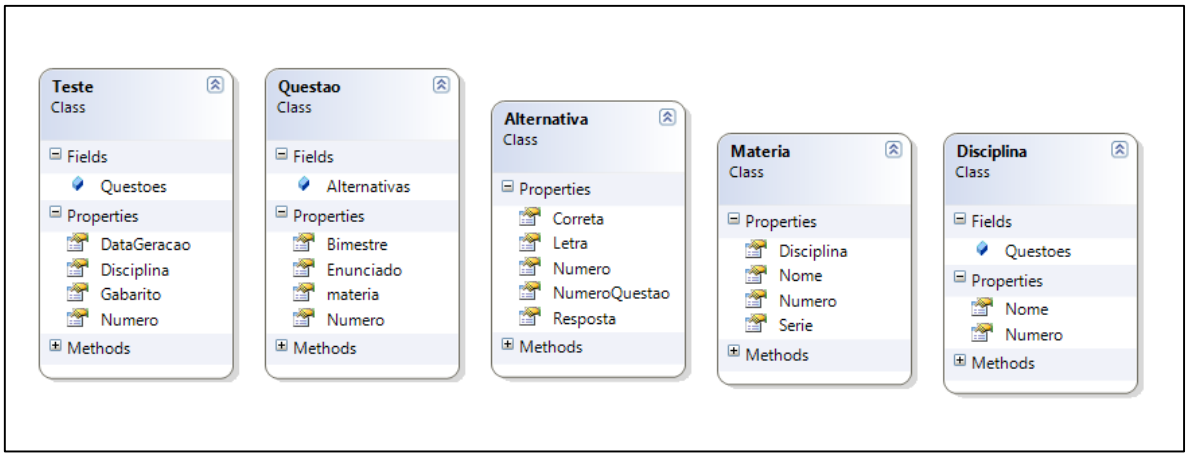

Figura 31 - Diagrama de classe (Protótipo)
Fonte: Próprio autor

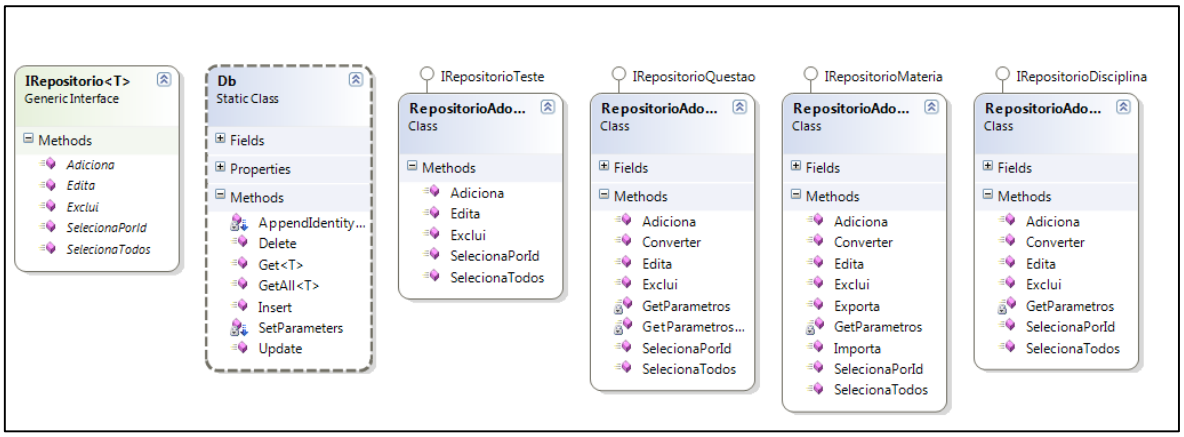

Figura 32 - Diagrama de classe Camada de acesso(Protótipo)
Fonte: Próprio autor

Diagrama de classe da camada de acesso ao banco de dados (Figura 32) criada seguindo o modelo repositório para melhor abstração da camada logica com a camada de acesso a dados.

4.2 Casos de Uso

O caso de uso (Figura 33) demonstra as principais funções do sistema, Inserção de Disciplinas, Matérias e questões, e por fim a geração da avaliação.

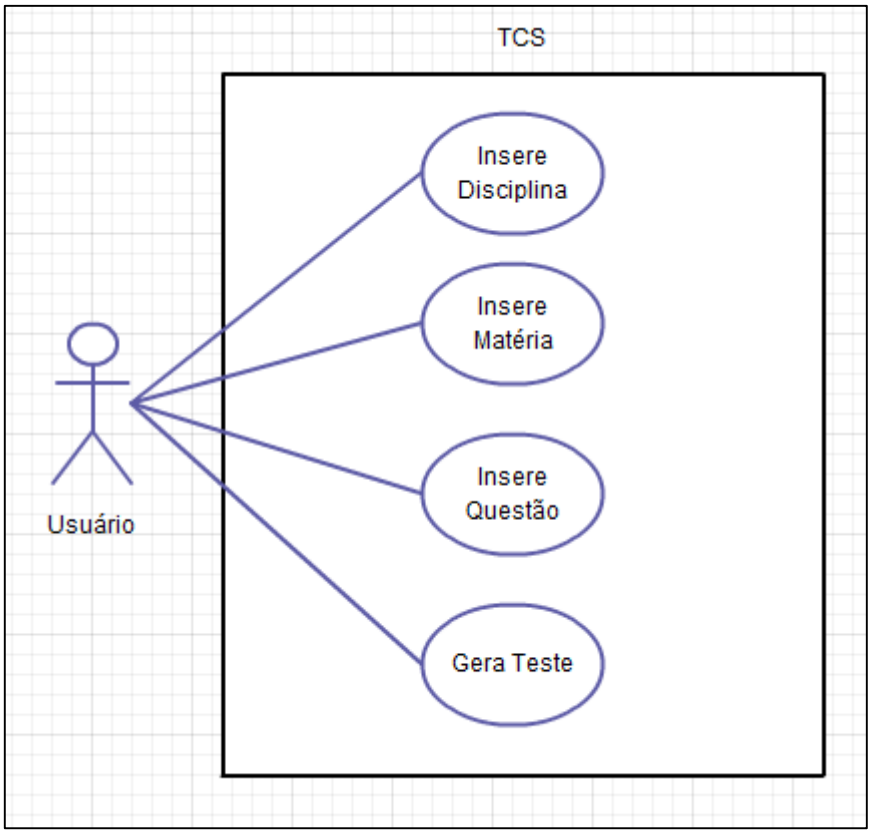

Figura 33 - Diagrama de Caso de Uso
Fonte: Próprio Autor

4.3 Interface

A tela principal do sistema (Figura 34) segue um padrão simples e de fácil entendimento. Mantendo sempre as mesmas funcionalidades para cada tipo de informação, Questões, Matérias e Disciplinas apresentam capacidade de criação, modificação e exclusão.

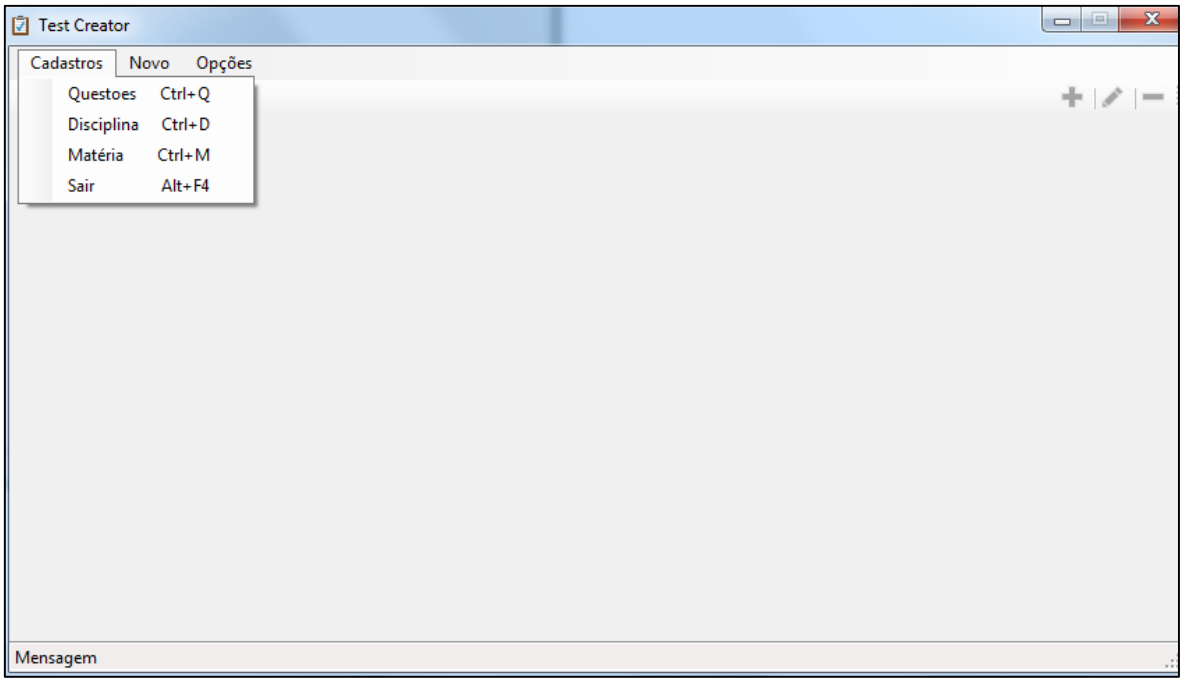

Figura 34- Tela Principal
Fonte: Próprio autor

Na inserção de questões (figura 35) o professor poderá escolher quantidade de alternativas, nível de dificuldade da questão, levando em conta qual questão ele, o professor, ache que tenha um maior peso na nota da avaliação.

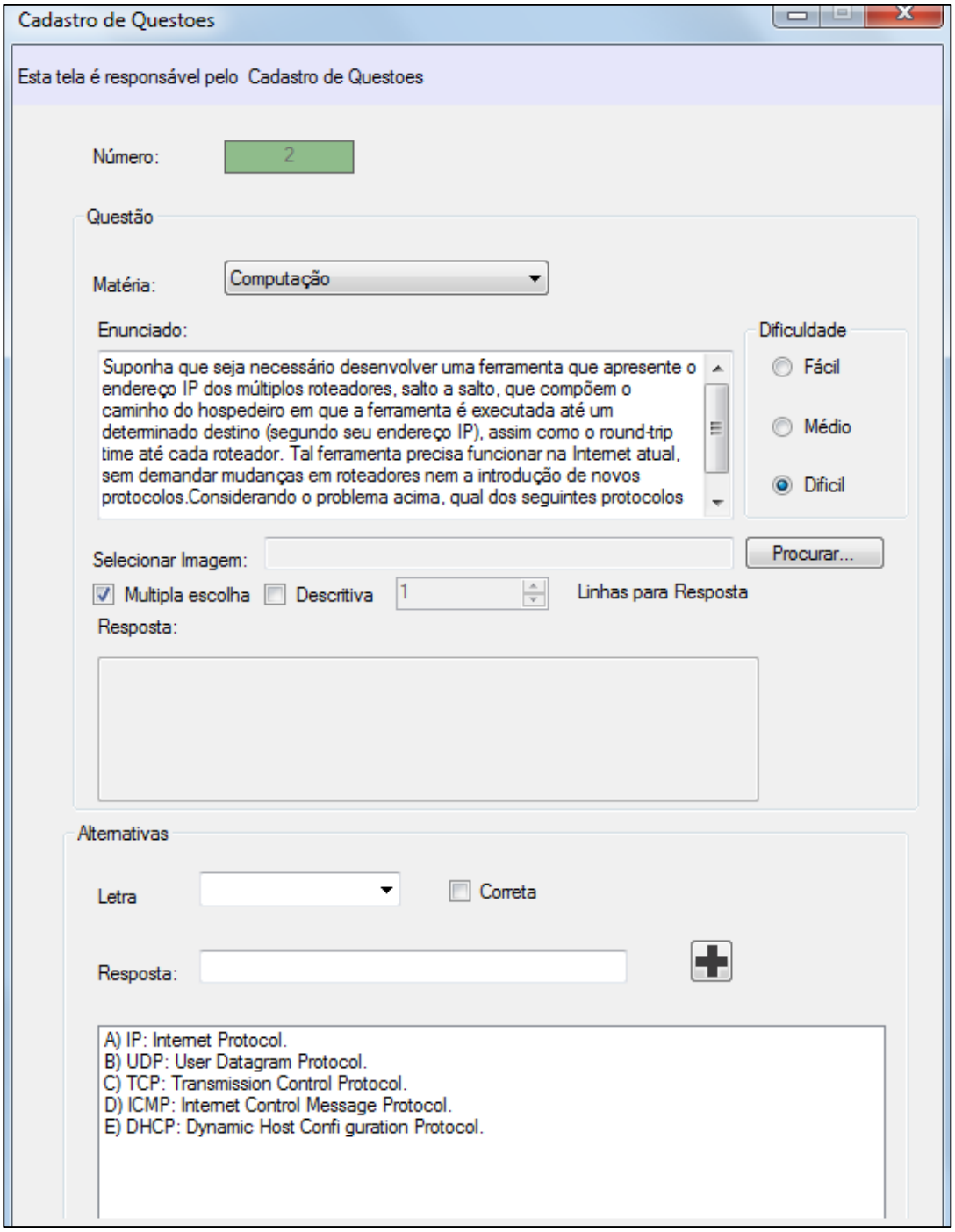

Figura 35: - Tela Cadastro de questões
Fonte: Próprio autor

Ao se obter um padrão para criação de avaliações que através do TCS seja de fácil reprodução e adaptação nos diversos níveis de ensino, visa facilitar o aprendizado e a preparação de alunos para as avaliações futuras.

A criação (Figura 36) de avaliações se dá de maneira simplificada pela interface, selecionando Matéria e disciplina para a prova.

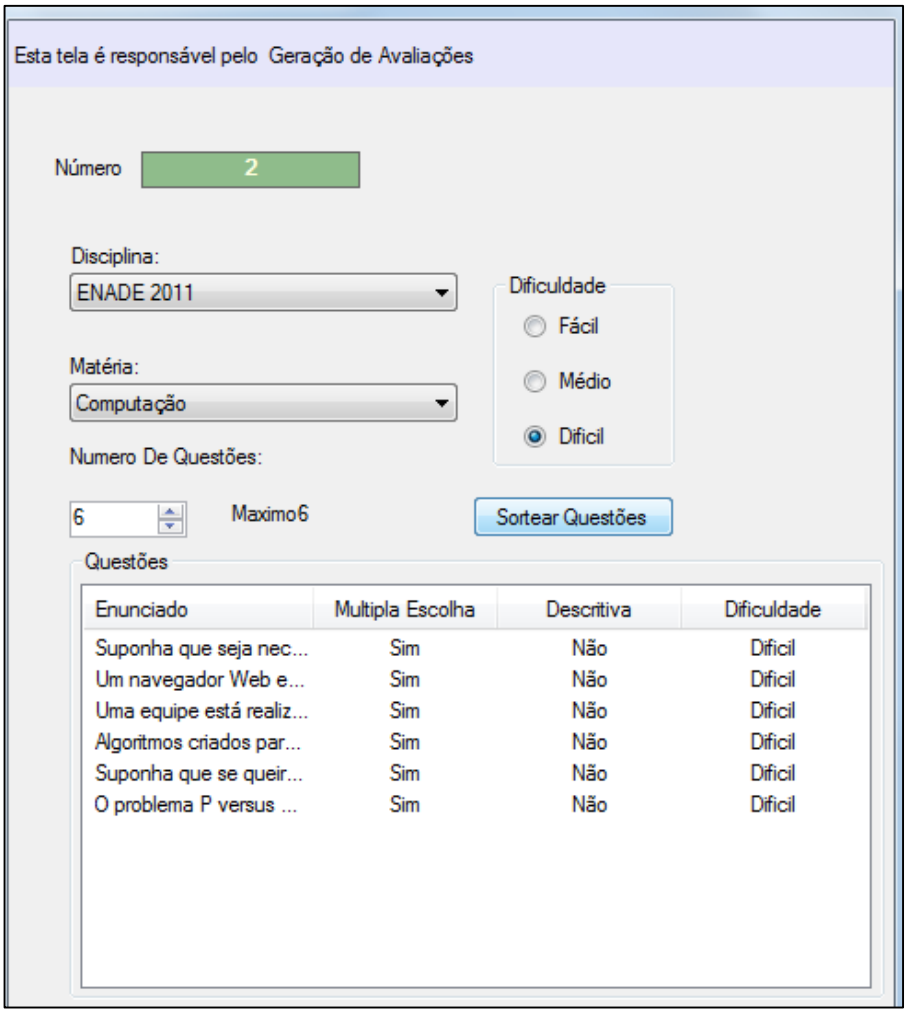

Figura 36: - Criação de Avaliação
Fonte: Próprio autor

Após criada a avaliação é mantida no banco de dados, podendo assim ser editada para mais uma vez ser utilizada ou para a geração do gabarito. A avaliação (figura 37) é gerada em formato.PDF no diretório de escolha do professor.

Aluno:_____ 10/11/2014

Questão 1

Suponha que seja necessário desenvolver uma ferramenta que apresente o endereço IP dos múltiplos roteadores, salto a salto, que compõem o caminho do hospedeiro em que a ferramenta é executada até um determinado destino (segundo seu endereço IP), assim como o round-trip time até cada roteador. Tal ferramenta precisa funcionar na Internet atual, sem demandar mudanças em roteadores nem a introdução de novos protocolos.Considerando o problema acima, qual dos seguintes protocolos representaria a melhor (mais simples e eficiente) solução?

A)IP: Internet Protocol.
B)UDP: User Datagram Protocol.
C)TCP: Transmission Control Protocol.
D)ICMP: Internet Control Message Protocol.
E)DHCP: Dynamic Host Confi guration Protocol.

Questão 2

Um navegador Web executa em um hospedeiro A, em uma rede de uma organização, e acessa uma página localizada de um servidor Web em um hospedeiro B, situado em outra rede na Internet. A rede em que A se situa conta com um servidor DNS local. Um profi ssional deseja fazer uma lista com a sequência de protocolos empregados e comparar com o resultado presentado por uma ferramenta de monitoramento executada no hospedeiro A. A lista assume que:

i) todas as tabelas com informações temporárias e caches estão vazias;
ii) o hospedeiro cliente está confi gurado com o endereço IP do servidor DNS local. Qual das sequências a seguir representa a ordem em que mensagens, segmentos e pacotes serão observados em um meio físico ao serem enviados pelo hospedeiro A?

A)ARP, DNS/UDP/IP, TCP/IP e HTTP/TCP/IP
B)ARP, DNS/UDP/IP, HTTP/TCP/IP e TCP/IP
C)DNS/UDP/IP, ARP, HTTP/TCP/IP e TCP/IP.
D)DNS/UDP/IP, ARP, TCP/IP e HTTP/TCP/IP.
E)HTTP/TCP/IP, TCP/IP, DNS/UDP/IP e ARP.

Figura 37: - Exemplo de avaliação
Fonte: Próprio autor

As avaliações ainda podem conter imagens ou questões descritivas, que quando for o caso, as linhas para resposta substituirão o espaço das alternativas.

V CONCLUSÃO

Utilizando padrões de projeto e linguagens atuais, o desenvolvimento de software fica muito mais simples de ser executado, garantindo um produto de qualidade e fácil manutenção tanto para o desenvolvedor quanto para o usuário final, que não terá problemas em usufruir do sistema.

Com a conclusão do desenvolvimento se espera obter um padrão para criação de avaliações levando em consideração os padrões de avaliação federal (POSCOMP e ENADE) sendo aplicado através do TCS (*Test Cretor System*) que será a plataforma de geração de avaliações, e que o sistema possa ser usado nos diversos níveis de ensino, tornando se assim uma ferramenta para facilitar e maximizar a organização de avaliações.

VI REFERÊNCIAS BIBLIOGRÁFICAS

BEIGHLEY, Lynn. **Use a Cabeça! SQL.** Alta Books, 2008

CARVALHO, Victorio Albani de; TEIXEIRA, Giovany Frossard**. Programação orientada a objetos: Curso técnico de informática**. Colatina: IFES, 2012. 134 p. Disponível em: <http://redeetec.mec.gov.br/images/stories/pdf/eixo_infor_comun/tec_inf/081112_progr_obj.pdf>. Acesso em: 30 março 2014.

Date,C J. **Introdução a sistemas de banco de dados 8° edição (tradução de daniel vieira).** Rio de janeiro: Elsevier ,2003

GAMMA, Erich et al**. Padrões de Projetos: soluções reutilizáveis de software orientado a objetos**. Porto Alegre: Bookman, 2000.

K19. **c# e orientação a objetos.** Disponível em: <http://www.k19.com.br/downloads/apostilas/dotnet/k19-k31-csharp-e-orientacao-a-objetos> Acesso em: 01 Abril 2014

PRESSMAN,Roger S. **Engenharia de software : uma abordagem profissional**. Porto Alegre: AMGH,2011.

KLINE, Kevin E. **SQL O Guia essencial - Manual de Referência do Profissional**. Alta Books, 2010

LASSALA, Claudio. **Programação Orientada a Objetos em .NET – Parte 1**. MSDN Magazine. 2014 Disponivel em <http://msdn.microsoft.com/pt-br/library/cc580626.aspx> Acesso em: 05 março 2014

MEYER, Bertrand. **Object-Oriented Software Construction.** Santa Barbara: ISE Inc., 2000. Disponível em: <http://www.dee.ufma.br/pub/UML/Bertrand%20Meyer%20-%20Object%20Oriented%20Software%20Construction%20%282Ed%29.pdf>. Acesso em: 30 março 2014.

Microsoft Corp.: **Introdução à linguagem C# e ao .NET Framework.** Disponivel em :<http://msdn.microsoft.com/pt-br/library/z1zx9t92.aspx>. Acesso em: 05 março 2014

O.K. Takai; I.C.Italiano; J.E. **Ferreira. Introdução a Banco de Dados.** DCC-IME-USP – Fevereiro – 2005. Disponível em: < http://www.ime.usp.br/~jef/apostila.pdf> Acesso em30 março 2014.

THEODORSON, G. A. & THEODORSON, A. G. **A modern dictionary of sociology**. London, Methuen, 1970.

WATSON, Karli. **Começando Visual C#.**Wiley Publishing, 2010

VII ANEXOS

```
using System;
using System.Drawing;
using System.Windows.Forms;
using Unifacvest.TCS.App.Controles.AboutForms;
using Unifacvest.TCS.App.Controles.Compartilhados;
using Unifacvest.TCS.App.Controles.ConfiguracoesForms;


namespace Unifacvest.TCS.App
{
    public partial class MainForm : Form
    {
        private static MainForm _instance;
        private IGerenciador _dataManager;
        private UserControl _control;

        public MainForm()
        {
            InitializeComponent();
            _instance = this;
        }



        public static MainForm Instance
        {
            get { return _instance ?? (_instance = new MainForm()); }
        }

        /// <summary>
        /// Carrega lista correspondente ao controle selecionado
        /// </summary>
        /// <param name="key"></param>
        private void LoadDataManager(GerenciadorEnum key)
        {
            try
            {
                IGerenciador newManager = GerenciadorFactory.GetManager(key);

                if (newManager != null && _dataManager != newManager)
                {
                    if (_control != null)
                    {

                        panControl.Controls.Clear();
                    }

                    _dataManager = newManager;
                    _control = _dataManager.GetControl();

                    _control.Width = panControl.Width;
                    _control.Height = panControl.Height;
                    _control.Dock = DockStyle.Fill;
```

```
                panControl.Controls.Add(_control);
                toolbar.Items["descricao"].Text = _dataManager.Descricao();

                var toolTip = _dataManager.GetToolTipMessage();
                if (toolTip != null)
                {
                    btnAdd.ToolTipText = toolTip.Adiciona;
                    btnRemove.ToolTipText = toolTip.Deleta;
                    btnEdita.ToolTipText = toolTip.Edita;
                    btnPesquisar.ToolTipText = toolTip.Pesquisar;
                }

                var actions = _dataManager.GetActionButtonEnabled();
                if (actions != null)
                {
                    btnAdd.Enabled = actions.Adiciona;
                    btnRemove.Enabled = actions.Remove;
                    btnEdita.Enabled = actions.Edita;
                    btnPesquisar.Enabled = actions.Pesquisa;
                    txtPesquisar.Enabled = actions.Pesquisa;

                }
            }
            toolbar.Enabled = _dataManager != null;
        }
        catch (Exception be)
        {
            MessageBox.Show(be.Message);
        }
    }

    /// <summary>
    /// carrega lista de quesotes
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void QuestoesToolStripMenuItemClick(object sender, EventArgs e)
    {
        LoadDataManager(GerenciadorEnum.Questoes);

    }

    /// <summary>
    /// carrega lista de disciplinas
    /// </summary>
    /// <param name="sender"></param>
    /// <param name="e"></param>
    private void DisciplinaToolStripMenuItemClick(object sender, EventArgs
e)
    {
        LoadDataManager(GerenciadorEnum.Disciplina);
    }


    /// <summary>
    /// Carrega lista de Materias
    /// </summary>
```

```
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void MatériaToolStripMenuItemClick(object sender, EventArgs e)
        {
            LoadDataManager(GerenciadorEnum.Materia);
        }


        private void BtnAddClick(object sender, EventArgs e)
        {
            try
            {
                _dataManager.AddData();
            }
            catch (Exception exc)
            {
                MessageBox.Show(exc.Message);
            }
        }

        private void BtnEditaClick(object sender, EventArgs e)
        {
            try
            {
                _dataManager.EditData();
            }
            catch (Exception exc)
            {
                MessageBox.Show(exc.Message);
            }
        }

        private void BtnRemoveClick(object sender, EventArgs e)
        {
            try
            {
                _dataManager.RemoveData();
            }
            catch (Exception exc)
            {
                MessageBox.Show(exc.Message);
            }
        }

        private void NovoTesteToolStripMenuItemClick(object sender, EventArgs e
)
        {
            LoadDataManager(GerenciadorEnum.Prova);
        }

        private void SairToolStripMenuItemClick(object sender, EventArgs e)
        {
            Close();
        }

        public void MostraMesagemStatus(string mesagem,Color alertColor)
        {
```

```
            lblStatus.Text = mesagem;
            lblStatus.ForeColor = alertColor;
        }

        private void ConfiguraçõesToolStripMenuItem1Click(object sender, EventA
rgs e)
        {
            var configuracao= new ConfiguracaoDialog();


            if(configuracao.ShowDialog()==DialogResult.OK)
            {
            }
        }


        private void AboutToolStripMenuItemClick(object sender, EventArgs e)
        {
           var aboutBox= new AboutBox();

            aboutBox.Show();
        }


        /// <summary>
        /// evento para pesquisa de itens
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void toolStripButton1_Click(object sender, EventArgs e)
        {
            try
            {
                _dataManager.Pesquisa(txtPesquisar.Text);
            }
            catch (Exception exc)
            {
                MessageBox.Show(exc.Message);
            }
        }

        /// <summary>
        /// retorna lista completa quando o campo esta vazio.
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void toolStripTextBox1_TextChanged(object sender, EventArgs e)
        {
            if (txtPesquisar.Text.Length == 0)
            {
                try
                {
                    _dataManager.Pesquisa(txtPesquisar.Text);
                }
                catch (Exception exc)
                {
                    MessageBox.Show(exc.Message);
```

```
                }
            }
        }


    }
}
using System;
using System.Collections.Generic;
using System.Drawing;
using System.IO;
using System.Text;
using Unifacvest.TCS.Modelo;
using Unifacvest.TCS.Repositorio.ADO;
using iTextSharp.text;
using iTextSharp.text.pdf;

namespace Unifacvest.TCS.App.GeradorPDF
{
    public class PDFFactory
    {
        /// <summary>
        /// gera o a prova em formato pdf
        /// </summary>
        /// <param name="questoes"></param>
        /// <param name="caminho"></param>
        /// <returns></returns>
        public bool CriarProvaPDf(List<Questao> questoes, string caminho)
        {

            //cria novo documento pdf
            var documento = new Document();
            var pdfWriter = PdfWriter.GetInstance(documento, new FileStream(cam
inho, FileMode.Create));
            pdfWriter.PageEvent = new EventosPDF();

            #region Info inst
            //parte fixa do PDF, pode ser configurado novos parametros se preci
so.
            var instuicao = new Paragraph("Centro Universitario UNIFACVEST") {
Alignment = Element.ALIGN_CENTER };
            var materia = new Paragraph("Matéria: " + questoes[0].Materia.Nome)
 { Alignment = Element.ALIGN_CENTER };
            var configuracao = new RepositorioAdoConfiguracao().SelecionaPorId(
1);
            var professor = new Paragraph("Professor: " + configuracao.Professo
r + "-" + configuracao.Email) { Alignment = Element.ALIGN_CENTER };

            #endregion

            var blankLine = new Paragraph(" ");
            var aluno = new Paragraph("Aluno:_____
_____ " + DateTime.Now.ToString("dd/MM/yyyy"));

            var fonte = new iTextSharp.text.Font { Color = new BaseColor(Color.
White) };
```

```
            documento.Open();

            //fixo
            documento.Add(instuicao);
            documento.Add(materia);
            documento.Add(professor);
            documento.Add(blankLine);
            documento.Add(aluno);

            int numero = 0;
            //adicionda cada questao ao arquivo pdf
            foreach (var questao in questoes)
            {
                numero++;
                var tabela = new PdfPTable(1) { HorizontalAlignment = 0, WidthP
ercentage = 100 };
                var cell = new PdfPCell(new Phrase("Questão " + numero, fonte))
 { BackgroundColor = new BaseColor(Color.Black) };
                tabela.AddCell(cell);

                documento.Add(blankLine);
                documento.Add(blankLine);
                documento.Add(tabela);
                var enunciado = new Paragraph(questao.Enunciado);
                enunciado.Alignment = Element.ALIGN_JUSTIFIED;
                documento.Add(enunciado);
                documento.Add(blankLine);

                //caso possua uma imagem , adiciona após a questao
                if (questao.IdImagem > 0)
                {
                    var f = new RepositorioAdoImagem().SelecionaPorId(questao.I
dImagem);
                    iTextSharp.text.Image img = iTextSharp.text.Image.GetInstan
ce(f.ImagemBytes);
                    img.Alignment = Element.ALIGN_CENTER;
                    img.ScalePercent(60);
                    img.Border = 15;
                    img.BorderColor = new BaseColor(Color.Black);
                    img.BorderWidth = 2;
                    img.ScaleAbsolute(200f, 200f);
                    documento.Add(img);




                }

                //define se é de multipla escolha
                if (questao.MultiplaEscolha)
                {
                    foreach (var alternativa in questao.Alternativas)
                    {
                        documento.Add(new Paragraph(alternativa.Letra + ")" + a
lternativa.Resposta));
```

```
                }
            }


            //define se é descritiva
            if (questao.Descritiva)
            {
                for (int i = 0; i < questao.LinhasResposta; i++)
                {

                    documento.Add(new Paragraph(Montalinha()));

                }
            }


        }


            documento.Close();

            return true;
        }

        /// <summary>
        /// Gera gabarito das questoes
        /// </summary>
        /// <param name="questoes"></param>
        /// <param name="caminho"></param>
        /// <returns></returns>
        public bool CriarGabaritoPDF(List<Questao> questoes, string caminho)
        {
            var documento = new Document();
            PdfWriter.GetInstance(documento, new FileStream(caminho, FileMode.C
reate));

            #region Info inst

            //parte fixa da prova.
            var instuicao = new Paragraph("Centro Universitario UNIFACVEST") {
Alignment = Element.ALIGN_CENTER };
            var materia = new Paragraph("Matéria: " + questoes[0].Materia.Nome)
 { Alignment = Element.ALIGN_CENTER };
            var configuracao = new RepositorioAdoConfiguracao().SelecionaPorId(
1);
            var professor = new Paragraph("Professor: " + configuracao.Professo
r + "-" + configuracao.Email) { Alignment = Element.ALIGN_CENTER };

            var fonteGabarito = new iTextSharp.text.Font { Color = new BaseColo
r(Color.Red) };
            var aviso = new Paragraph("GABARITO", fonteGabarito) { Alignment =
Element.ALIGN_CENTER };

            #endregion

            var blankLine = new Paragraph(" ");
```

```
            var fonte = new iTextSharp.text.Font { Color = new BaseColor(Color.
White) };

            documento.Open();

            //fixo
            documento.Add(instuicao);
            documento.Add(materia);
            documento.Add(professor);
            documento.Add(aviso);
            documento.Add(blankLine);


            int numero = 0;
            foreach (var questao in questoes)
            {
                numero++;
                var tabela = new PdfPTable(1) { HorizontalAlignment = 0, WidthP
ercentage = 100 };
                var cell = new PdfPCell(new Phrase("Questão " + numero, fonte))
 { BackgroundColor = new BaseColor(Color.Black) };
                tabela.AddCell(cell);

                documento.Add(blankLine);
                documento.Add(blankLine);
                documento.Add(tabela);
                documento.Add(new Paragraph(questao.Enunciado) { Alignment = 3
});
                documento.Add(blankLine);

                if (questao.MultiplaEscolha)
                {
                    foreach (var alternativa in questao.Alternativas)
                    {
                        documento.Add(alternativa.Correta
                                        ? new Paragraph(alternativa.Letra + "
)" + alternativa.Resposta, fonteGabarito)
                                        : new Paragraph(alternativa.Letra + "
)" + alternativa.Resposta));
                    }
                }

                if (questao.Descritiva)
                {


                    documento.Add(new Paragraph(questao.Resposta, fonteGabarito
) { Alignment = 3 });


                }


            }


            documento.Close();
```

```
            return true;
        }

        /// <summary>
        /// adiciona numero de linhas a questao descritiva
        /// </summary>
        /// <returns></returns>
        private string Montalinha()
        {
            StringBuilder builder = new StringBuilder();

            for (int i = 0; i < 78; i++)
            {
                builder.Append("_");
            }

            return builder.ToString();
        }
    }
}
```

```
using Unifacvest.TCS.App.Controles.DisciplinaForms;
using Unifacvest.TCS.App.Controles.MateriaForms;
using Unifacvest.TCS.App.Controles.ProvaForms;
using Unifacvest.TCS.App.Controles.QuestoesForms;

namespace Unifacvest.TCS.App.Controles.Compartilhados
{
    public class GerenciadorFactory
    {
        /// <summary>
        /// Carrega o gerenciador apartir da chave informada
        /// </summary>
        /// <param name="key"></param>
        /// <returns></returns>
        public static IGerenciador GetManager(GerenciadorEnum key)
        {

            switch (key)
            {
                case GerenciadorEnum.Disciplina:
                   return new DisciplinaGerenciadorImpl();
                    break;


                case GerenciadorEnum.Materia:
                   return new MateriaGerenciadorImpl();
                    break;

                case GerenciadorEnum.Questoes:
                    return new QuestaoGerenciadorImpl();
                   break;
```

```
                case GerenciadorEnum.Prova:
                    return new ProvaGerenciadorImpl();
                   break;
            }


            return null;

        }
    }
}
using System;
using System.Windows.Forms;
using Unifacvest.TCS.App.Controles.Compartilhados;
using Unifacvest.TCS.Modelo;
using Unifacvest.TCS.Repositorio.ADO;

namespace Unifacvest.TCS.App.Controles.ProvaForms
{
    internal class ProvaGerenciadorImpl : GerenciadorBase<Prova>, IGerenciador
    {
        public ProvaGerenciadorImpl()
            : base(new RepositorioAdoProva(), new ProvaControl())
        {
        }

        public void AddData()
        {
            var dialog = new ProvaDialog(Descricao()) {Prova = new Prova()};
            if (dialog.ShowDialog() == DialogResult.OK)
            {
                Repositorio.Adiciona(dialog.Prova);
                RefreshAll();
            }
        }

        public void RemoveData()
        {
            var numero = Controle.GetId<int>();
            Prova prova = Repositorio.SelecionaPorId(numero);

            if (prova == null)
            {
                MessageBoxExtension.Show(
                    "Nenhuma Prova selecionada. Selecione uma Prova antes de so
licitar a exclusão",
                    MessageEnum.Warning);
                return;
            }
            if (MessageBoxExtension.ShowQuestion("Deseja remover a Prova seleci
onada?") == DialogResult.Yes)
            {
                try
                {
                    Repositorio.Exclui(prova);
                    RefreshAll();
```

```
                }
                catch (Exception e)
                {
                    MessageBoxExtension.Show(e.Message, MessageEnum.Error);
                }
            }
        }

        public void EditData()
        {
            var numero = Controle.GetId<int>();

            Prova teste = Repositorio.SelecionaPorId(numero);

            if (teste == null)
            {
                MessageBoxExtension.Show(
                    "Nenhuma Prova selecionado. Selecionar uma Prova antes de s
olicitar uma edição",
                    MessageEnum.Warning);
                return;
            }

            var dialog = new ProvaDialog(Descricao()) {Prova = teste};
            if (dialog.ShowDialog(Controle) == DialogResult.OK)
            {
                Repositorio.Edita(dialog.Prova);
                RefreshAll();
            }
        }

        public string Description()
        {
            return "Gerar Novo Teste";
        }

        public MensagemToolTip GetToolTipMessage()
        {
            return new MensagemToolTip
                       {Adiciona = "Adiciona um Teste", Edita = "Edita um Teste
", Deleta = "Remove um Teste"};


        }

        public ActionButtonEnabled GetActionButtonEnabled()
        {
            return new ActionButtonEnabled {Adiciona = true, Remove = true, Edi
ta = true};


        }

        public string Descricao()
        {
            return " Geração de Avaliações";
        }
```

```
    }
}


using System;
using System.Drawing;
using System.Windows.Forms;

namespace Unifacvest.TCS.App.Controles.Compartilhados
{
    public partial class FormModelo : Form
    {
        protected bool _configurando;
        public FormModelo()
        {
            InitializeComponent();

        }



        protected string DescricaoTela
        {
            set
            {
                labelTitulo.Text += value;
                Text += value;
            }
        }

        /// <summary>
        /// Desabilita botao "ok" caso seja necessario
        /// </summary>
        protected void DesabilitaBotaoOk()
        {
            btnOK.Enabled = false;
        }


        /// <summary>
        /// abilita botao "ok" quando necessario
        /// </summary>
        protected void HabilitaBotaoOk()
        {
            btnOK.Enabled = true;
        }


        /// <summary>
        /// Adiciona o vento de mudança de valor ao controle.
        /// </summary>
        /// <param name="controles"></param>
        protected void RegistraControles(params Control[] controles)
        {
            foreach (Control item in controles)
            {
                if (item is RadioButton)
```

```
                {
                    ((RadioButton)item).CheckedChanged += ValueChangedControl;
                }
                if (item is ListBox)
                {
                    ((ListBox)item).TextChanged += ValueChangedControl;
                }
                if (item is NumericUpDown)
                {
                    ((NumericUpDown)item).ValueChanged += ValueChangedControl;
                }
                if (item is TextBox)
                {
                    ((TextBox)item).TextChanged += ValueChangedControl;
                }
                if (item is ComboBox)
                {
                    ((ComboBox)item).TextChanged += ValueChangedControl;
                }
                if (item is CheckBox)
                {
                    ((CheckBox)item).CheckStateChanged += ValueChangedControl;
                }
                if (item is ListBox)
                {
                    ((ListBox)item).TextChanged += ValueChangedControl;
                }

            }
        }

        /// <summary>
        /// limpa Menssagem de status na form principal
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void BaseDialog_FormClosing(object sender, FormClosingEventArgs
e)
        {
            MainForm.Instance.MostraMesagemStatus(string.Empty,Color.Black);
        }

        /// <summary>
        /// evento para verificação de modificaçoes
        /// </summary>
        /// <param name="sender"></param>
        /// <param name="e"></param>
        private void ValueChangedControl(object sender, EventArgs e)
        {
            ModificacaoChek();
        }

        protected virtual void ModificacaoChek()
        {
        }

    }
```

```
}
```

```
using System;
using System.Drawing;
using Unifacvest.TCS.Modelo;

namespace Unifacvest.TCS.App.Controles.DisciplinaForms
{
    public partial class DisciplinaDialog : Compartilhados.FormModelo
    {
        private Disciplina _disciplina;
        public DisciplinaDialog(string descricao)
        {

            InitializeComponent();
            RegistraControles(txtNome, txtNumero);
            DescricaoTela = descricao;
            Size = new Size(Size.Width-50 , Size.Height - 580);

        }

        public Disciplina Disciplina
        {
            get { return _disciplina; }
            set
            {
                _configurando = true;
                _disciplina = value;

                txtNumero.Text = _disciplina.Id.ToString();
                txtNome.Text = _disciplina.Nome;

                _configurando = false;
                ModificacaoChek();
            }
        }

        protected override void ModificacaoChek()
        {
            if (_configurando)
                return;

            _disciplina.Nome = txtNome.Text;
            _disciplina.Id = int.Parse(txtNumero.Text);

            try
            {
                if (_disciplina.Valida())
                {
                    MainForm.Instance.MostraMesagemStatus("A disciplina já pode
 ser gravada");

                    HabilitaBotaoOk();
```

```
                }
            }
            catch (Exception ex)
            {
                DesabilitaBotaoOk();
                MainForm.Instance.MostraMesagemStatus(ex.Message);
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using Unifacvest.TCS.Modelo;
using Unifacvest.TCS.Repositorio.ADO;

namespace Unifacvest.TCS.App.Controles.MateriaForms
{
    public partial class MateriaDialog : Compartilhados.FormModelo
    {
        private Materia _materia;
        public MateriaDialog(string descricao)
        {
            InitializeComponent();
            RegistraControles(txtNumero, txtNome, cmbDisciplina);
            Size = new Size(Size.Width, Size.Height - 550);
            DescricaoTela = descricao;
            LoadDisciplinas();

        }


        public Materia Materia
        {
            get { return _materia; }
            set
            {
                _configurando = true;
                _materia = value;

                txtNumero.Text = _materia.Id.ToString();
                txtNome.Text = _materia.Nome;
                if (_materia.Disciplina != null)
                {
                    cmbDisciplina.Text = _materia.Disciplina.ToString();
                }


                _configurando = false;
                ModificacaoChek();
            }
```

```
        }
        protected override void ModificacaoChek()
        {
            if (_configurando)
                return;

            _materia.Nome = txtNome.Text;
            _materia.Id = int.Parse(txtNumero.Text);
            _materia.Disciplina = cmbDisciplina.SelectedItem as Disciplina;

            try
            {
                _materia.Valida();

                MainForm.Instance.MostraMesagemStatus("A disciplina já pode ser
 gravada");

                HabilitaBotaoOk();
            }
            catch (Exception ex)
            {
                DesabilitaBotaoOk();
                MainForm.Instance.MostraMesagemStatus(ex.Message);
            }
        }

        private void LoadDisciplinas()
        {
            List<Disciplina> disciplinas = new RepositorioAdoDisciplina().Selec
ionaTodos();

            cmbDisciplina.Items.Clear();
            foreach (var item in disciplinas.Where(item => item != null))
            {
                cmbDisciplina.Items.Add(item);
            }
        }
    }
}
```

```
using System;
using System.Collections.Generic;

namespace Unifacvest.TCS.Modelo
{
    public class Prova
    {
        public int Id { get; set; }

        public DateTime DataGeracao { get; set; }

        public int Dificuldade { get; set; }
```

```
        public List<Questao> Questoes;

        public Materia Materia { get; set; }

        public Disciplina Disciplina { get; set; }

        public bool ValidaCampos()
        {
            if (Materia == null)
            {
                throw new Exception("Matéria não selecionada !");
            }

            if (Disciplina == null)
            {
                throw new Exception("Disciplia não selecionada !");
            }
            if (Dificuldade == 0)
            {
                throw new Exception("Dificuldade não selecionada !");
            }

            return true;
        }
    }
}
```

```
using System;

namespace Unifacvest.TCS.Modelo
{
    public class Materia
    {
        public int Id { get; set; }

        public string Nome { get; set; }

        public Disciplina Disciplina { get; set; }

        public override string ToString()
        {
            return Nome;
        }

        public void Valida()
        {
            if (string.IsNullOrEmpty(Nome))
            {
                throw new Exception("Nome não pode estar vazio !");
            }

            if (this.Disciplina == null)
            {
```

```
                throw new Exception("Disciplia não selecionada !");
            }
        }
    }
}


namespace Unifacvest.TCS.Modelo
{
   public class Imagem
    {

       public int Id { get; set; }

       public string Caminho { get; set; }

       public byte[] ImagemBytes { get; set; }

    }
}


using System;
using System.Collections.Generic;

namespace Unifacvest.TCS.Modelo
{
    public class Disciplina
    {
        public int Id { get; set; }

        public string Nome { get; set; }

        public List<Questao> Questoes = new List<Questao>();

        public override string ToString()
        {
            return Nome;
        }

        public bool Valida()
        {
            if (string.IsNullOrEmpty(this.Nome))
            {
                throw new Exception("Nome não pode estar vazio!");
            }

            return true;
        }
    }
}


using System;
using System.Collections.Generic;
using System.Linq;
```

```
using System.Text;

namespace Unifacvest.TCS.Modelo
{
    public class Configuracao
    {
        public int Id { get; set; }

        public string Professor { get; set; }

        public string Email { get; set; }

        public int ProvedorSQL { get; set; }


    }
}
```

De: Maria José de Paula Castanho <zeza@unicentro.br>

Data: 10 de novembro de 2014 21:48

Assunto: [RECEN] Agradecimento pela Submissão

Para: Carlos Guilherme Coelho <guilherme.coelho04@gmail.com>

Carlos Guilherme Coelho,

Agradecemos a submissão do seu manuscrito "Sistema de Geração de Avaliações Automatizado" para RECEN - Revista Ciências Exatas e Naturais. Através da interface de administração do sistema, utilizado para a submissão, será possível acompanhar o progresso do documento dentro do processo editorial, bastanto logar no sistema localizado em:

URL do Manuscrito:

http://revistas.unicentro.br/index.php/RECEN/author/submission/3187

Login: carlos-coelho

Em caso de dúvidas, envie suas questões para este email. Agradecemos mais

uma vez considerar nossa revista como meio de transmitir ao público seu

trabalho.

Maria José de Paula Castanho

RECEN - Revista Ciências Exatas e Naturais

Sistema de Geração de Avaliações Automatizado

Resumo: Este trabalho apresenta um sistema gerador de avaliações, com o objetivo de automatizar e padronizar a geração de provas. Para atingir tal meta foram realizadas pesquisas bibliográficas referentes a padrões em sistemas computacionais, orientação a objetos e persistência em banco de dados. O desenvolvimento de aplicações que automatizem diversos processos na vida dos usuários é cada vez maior, variando de pequenos aplicativos de previsão do tempo até sistemas de gestão financeira. A padronização das avaliações facilita a construção, reutilização, dificuldade e aplicação de um teste pelo professor, e norteia o aluno à continuidade e formato da avaliação, pois uma vez conhecido o padrão, não haverá surpresa ao realizar uma avaliação que segue a mesma metodologia. Com o sistema gerador de avaliação, o professor padrão de resposta necessita apenas cadastrar as questões referentes a sua disciplina e depois gerar uma avaliação com quantidade, nível e formato de questões que desejar.

Palavras chave: Gerador de avaliações, Sistemas computacionais, padrão de resposta.

Abstract: This paper presents a system for generating tests aiming, to standardize and automate the generation of tests. To achieve this goal literature searches relating to standards in computing systems, object orientation and persistence in the database were performed. The development of applications that automate various processes in the life of users is increasing, ranging from small applications of weather forecasting to financial management systems. The standardization of ratings facilitates the construction, reuse, implementation and difficulty of a test by the teacher guides the student and the continuity

and shape of the test, once known as the standard, there will be no surprise to an assessment that follows the same methodology. With the generator rating system, the response pattern teacher needs only registering concerns about their discipline and then generate an evaluation with quantity, level and format of questions that you want.

Keywords: Generator reviews, computer systems, response pattern.

1 Introdução

Desde a criação dos primeiros computadores na década de 40, os softwares criados tinham como objetivo, ajudar o usuário a completar uma tarefa mais rapidamente. Quando uma nova tecnologia é bem-sucedida na resolução de um problema, rapidamente se torna uma tendência que será utilizada pela maioria dos desenvolvedores na criação de software e se torna o próximo nível a ser modificado pela própria tecnologia (PRESSMAN, 2010). As tecnologias evoluíram e os problemas que devem ser solucionados seguem um ritmo de evolução ainda maior. Cada ano dezenas de milhares de novos usuários se lançam em busca de sistemas que ofereçam soluções rápidas e objetivas para os mais diversos problemas.

O senso comum ensina “o ser humano teme o desconhecido”, partindo desse ponto. Observamos que a metodologia de avaliação feita por um professor se difere dos demais professores e assim respectivamente, porque após criar uma zona de segurança o sistema será o mesmo sem mudanças. Porem um acadêmico que tem diversas aulas, sobre variados temas durante sua graduação tem que se adequar a cada sistema, dificultando assim sua preparação para uma avaliação.

A padronização no formato de provas cria uma segurança ao aluno que já sabe o que esperar sobre a avaliação e ajuda ao professor manter o equilíbrio entre dificuldade e conteúdo cobrado na avaliação.

2 Desenvolvimento

Para o desenvolvimento de qualquer sistema computacional é preciso identificar as necessidades e o perfil do usuário final, adaptando assim suas funcionalidades, requisitos e o hardware necessários para a completa utilização do sistema.

Para criação do TCS (Test Creator System), será utilizada a linguagem C# através do Visual Estudio, que se torna perfeito para criação de sistemas desktop com é o caso do TSC.O principio básico do sistema é a criação de provas automaticamente, com uma interface simples e menus intuitivos o sistema será de fácil utilização independentemente da intimidade com o uso do computador do usuário.

A tela principal do sistema (Figura 1) segue um padrão simples e de fácil entendimento. Mantendo sempre as mesmas funcionalidades para cada tipo de informação, Questões, Matérias e Disciplinas apresentam capacidade de criação, modificação e exclusão.

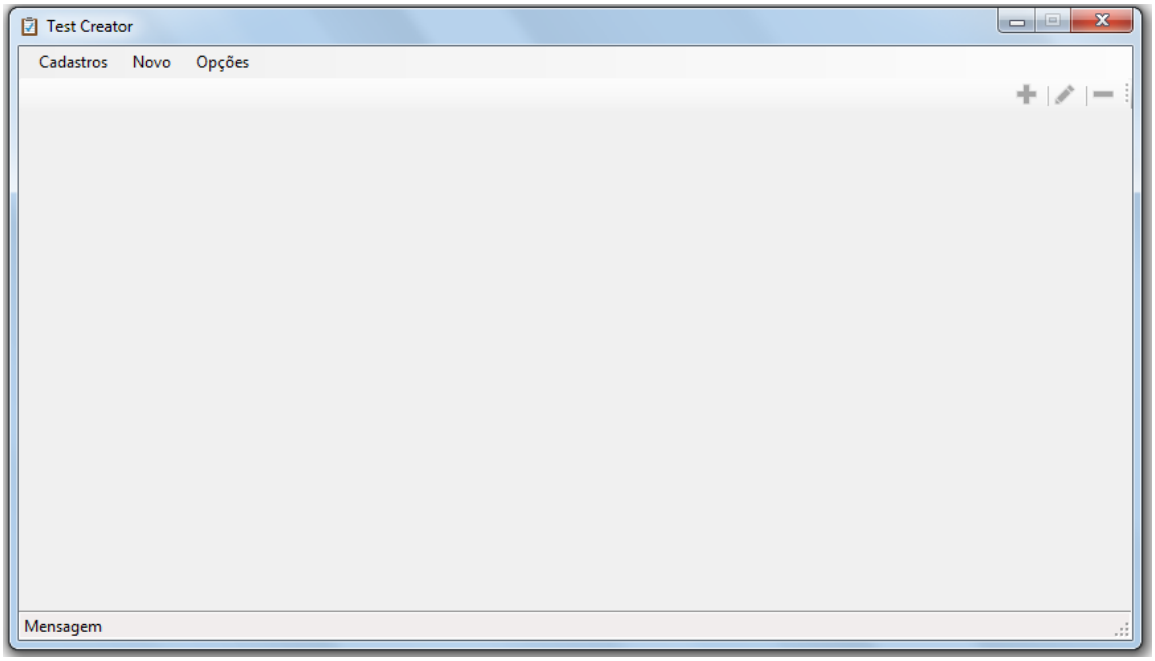

Figura 1 : Tela Principal

Na inserção de questões (figura 2) o professor poderá escolher quantidade de alternativas, nível de dificuldade da questão, levando em conta qual questão ele, o professor, ache que tenha um maior peso na nota da avaliação.

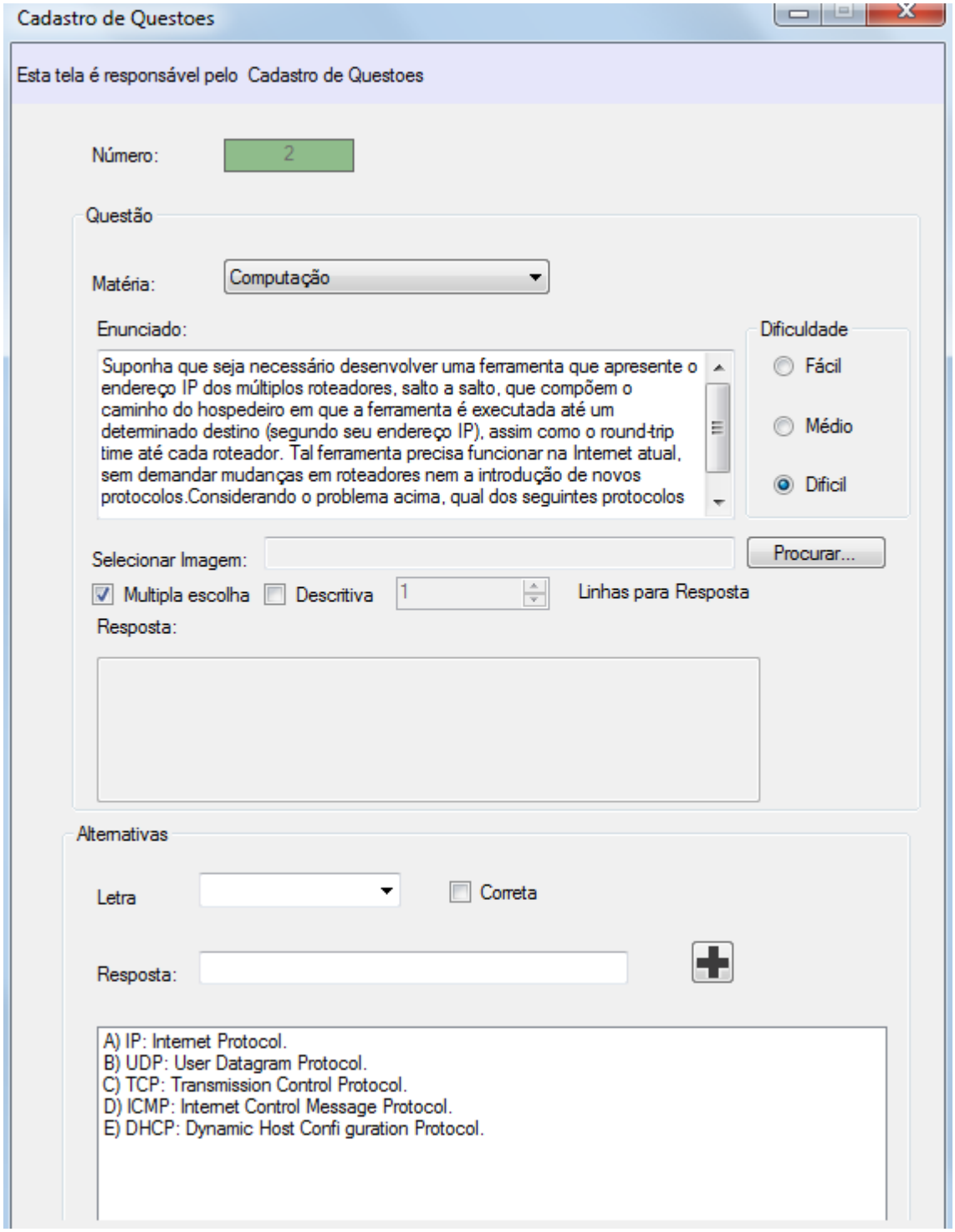

Figura 2 : Tela de adição de questões

3 Conclusão

Ao se obter um padrão para criação de avaliações que através do TCS seja de fácil reprodução e adaptação nos diversos níveis de ensino, visa facilitar o aprendizado e a

preparação de alunos para as avaliações futuras. A criação (Figura 3) de avaliações se da de maneira simplificada pela interface, selecionando Matéria e disciplina para a prova.

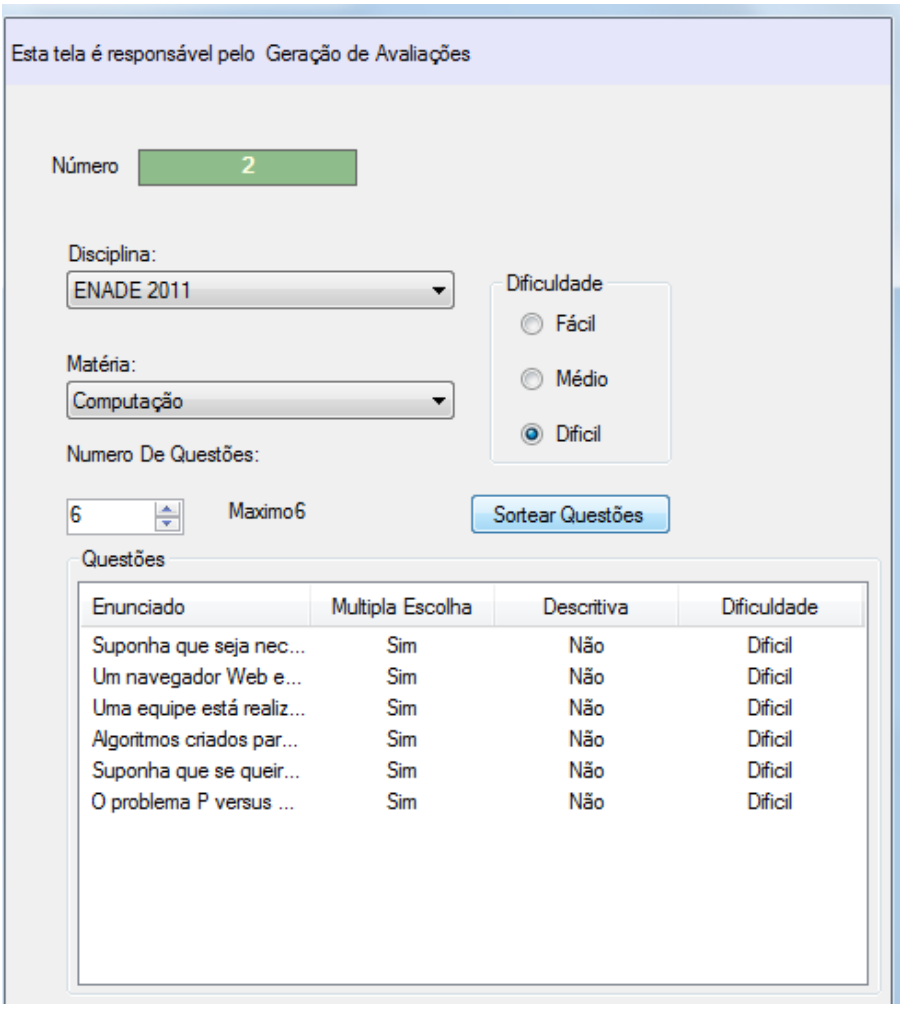

Figura 3:Exemplo de criação de Avaliação

Após criada a avaliação é mantida no banco de dados, podendo assim ser editada para mais uma vez ser utilizada ou para a geração do gabarito. A avaliação (figura 4) é gerada em formato.PDF no diretório de escolha do professor.

Aluno:_____ 10/11/2014

Questão 1

Suponha que seja necessário desenvolver uma ferramenta que apresente o endereço IP dos múltiplos roteadores, salto a salto, que compõem o caminho do hospedeiro em que a ferramenta é executada até um determinado destino (segundo seu endereço IP), assim como o round-trip time até cada roteador. Tal ferramenta precisa funcionar na Internet atual, sem demandar mudanças em roteadores nem a introdução de novos protocolos.Considerando o problema acima, qual dos seguintes protocolos representaria a melhor (mais simples e eficiente) solução?

A)IP: Internet Protocol.
B)UDP: User Datagram Protocol.
C)TCP: Transmission Control Protocol.
D)ICMP: Internet Control Message Protocol.
E)DHCP: Dynamic Host Confi guration Protocol.

Questão 2

Um navegador Web executa em um hospedeiro A, em uma rede de uma organização, e acessa uma página localizada de um servidor Web em um hospedeiro B, situado em outra rede na Internet. A rede em que A se situa conta com um servidor DNS local. Um profi ssional deseja fazer uma lista com a sequência de protocolos empregados e comparar com o resultado presentado por uma ferramenta de monitoramento executada no hospedeiro A. A lista assume que:

i) todas as tabelas com informações temporárias e caches estão vazias;
ii) o hospedeiro cliente está confi gurado com o endereço IP do servidor DNS local. Qual das sequências a seguir representa a ordem em que mensagens, segmentos e pacotes serão observados em um meio físico ao serem enviados pelo hospedeiro A?

A)ARP, DNS/UDP/IP, TCP/IP e HTTP/TCP/IP
B)ARP, DNS/UDP/IP, HTTP/TCP/IP e TCP/IP
C)DNS/UDP/IP, ARP, HTTP/TCP/IP e TCP/IP.
D)DNS/UDP/IP, ARP, TCP/IP e HTTP/TCP/IP.
E)HTTP/TCP/IP, TCP/IP, DNS/UDP/IP e ARP.

Figura 4: Exemplo de avaliação

As avaliações ainda podem conter imagens ou questões descritivas, que quando for o caso, as linhas para resposta substituirão o espaço das alternativas.

REFERÊNCIAS

[1] BEIGHLEY, Lynn. Use a Cabeça! SQL. Alta Books, 2008

[2] CARVALHO, Victorio Albani de; TEIXEIRA, Giovany Frossard. Programação orientada a objetos: Curso técnico de informática. Colatina: IFES, 2012. 134 p. Disponível em: <http://redeetec.mec.gov.br/images/stories/pdf/eixo_infor_comun/tec_inf/081112_progr_obj.pdf>. Acesso em: 30 março 2014.

[3] Date,C J. Introdução a sistemas de banco de dados 8° edição (tradução de daniel vieira). Rio de janeiro :Elsevier ,2003

[4] GAMMA, Erich et al. Padrões de Projetos: soluções reutilizáveis de software orientado a objetos. Porto Alegre: Bookman, 2000.

[5] K19.c# e orientação a objetos. Disponível em : <http://www.k19.com.br/downloads/apostilas/dotnet/k19-k31-csharp-e-orientacao-a-objetos> Acesso em: 01 Abril 2014

[6] PRESSMAN,Roger S. Engenharia de software : uma abordagem profissional. Porto Alegre:AMGH,2011.

[7] KLINE, Kevin E. SQL O Guia essencial - Manual de Referência do Profissional. Alta Books, 2010

[8] LASSALA, Claudio. Programação Orientada a Objetos em .NET – Parte 1 .MSDN Magazine. 2014 Disponivel em <http://msdn.microsoft.com/pt-br/library/cc580626.aspx> Acesso em: 05 março 2014.

[9] Questões de exemplo retiradas do ENADE 2011.