Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

EDUARDO NICOLINI SODRE DA SILVA

DESENVOLVIMENTO DO FRAMEWORK JAVA-FÁCIL

Assis
2011

Av. Getúlio Vargas, 1200 – Vila Nova Santana – Assis – SP – 19807-634
Fone/Fax: (0XX18) 3302 1055 homepage: www.fema.edu.br

Fundação Educacional do Município de Assis
Instituto Municipal de Ensino Superior de Assis
Campus "José Santilli Sobrinho"

EDUARDO NICOLINI SODRE DA SILVA

DESENVOLVIMENTO DO FRAMEWORK JAVA-FÁCIL

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis IMESA como requisito do Curso de Graduação.

Orientador: Luiz Carlos Begosso

Área de Concentração: Informática

.

Assis
2011

FICHA CATALOGRÁFICA

SODRE, Eduardo

Desenvolvimento do Framework Java - Fácil / Eduardo Nicolini Sodre da Silva, Fundação Educacional do Município de Assis – FEMA – Assis, 2011.

44p.

Orientador: Luiz Carlos Begosso.

Trabalho de Conclusão de Curso – Instituto Municipal de Ensino Superior de Assis – IMESA.

1. Framework. 2. Java.

CDD:

Biblioteca da FEMA

DESENVOLVIMENTO DO FRAMEWORK JAVA-FÁCIL

EDUARDO NICOLINI SODRE DA SILVA

Trabalho de Conclusão de Curso apresentado ao Instituto Municipal de Ensino Superior de Assis IMESA como requisito do Curso de Graduação, analisado pela seguinte comissão examinadora:

Orientador: Luiz Carlos Begosso

Analisador (1): Marisa Atsuko Nitto

.

Assis
2011

DEDICATÓRIA

Dedico este trabalho primeiramente ao Senhor Deus, que através do seu infinito amor me guardou, me deu forças e sabedoria todos os dias desta jornada. Com honra, dedico também a meu pai José Sodre da Silva e a minha mãe Rosangela Nicolini Sodre da Silva que com muita paciência e amor sempre me deu suporte e me orientou a seguir pelo caminho da retidão de caráter.

AGRADECIMENTOS

Agradeço primeiramente a Deus, pelas oportunidades e condições que me foram concedidas.

Aos meus pais que sempre me apoiaram nos estudos, me dando força e compreendendo os momentos em que estive ausente por causa dos estudos.

Aos amigos do Curso, que eu não poderia esquecer, que estiveram comigo dividindo as lutas e também as vitórias a cada etapa concluída, dos quais levo comigo lições de vida e boas lembranças.

Ao professor orientador Luiz Carlos Begosso pelo apoio e contribuição, bem como a todos os professores que de uma forma ou outra, me ajudaram a ultrapassar esta etapa, obrigado pela riqueza de suas palavras e pela aprendizagem proporcionada.

Quando pensamos nos agradecimentos sempre esquecemos de alguns atores participantes dessa nossa história, um grandioso passo em nossas vidas, assim, deixamos registrados o nosso sincero obrigado àqueles que de alguma forma aturam nessa conquista.

RESUMO

O presente trabalho refere-se ao desenvolvimento de um *framework* de apoio à criação de sistemas em ambiente Desktop e Web, que utilizam banco de dados para armazenamento de suas informações, com foco nas operações básicas de CRUD (*Create*, *Retrieve*, *Update*, *Delete*), Menu Principal, consultas simples, consultas avançadas, relatórios e criação do banco de dados.

Para o ambiente desktop será gerado o código na linguagem JAVA para diversos banco de dados, no ambiente web será gerado o código na linguagem PHP para o banco de dados Mysql.

Nesse sentido o trabalho inicia-se com uma revisitação da bibliografia acerca dos conceitos, padrões e metodologias das linguagens de programação e ferramentas utilizadas neste.

Palavras chave: *Framework*, Java, Hibernate, PHP, Desenvolvimento WEB

ABSTRACT

The present work refers to developing a *framework* to support the creation of systems on desktop and web environment, using the database to store your information, focusing on the basics of CRUD (Create Retrieve Update Delete), Main Menu , simple queries, advanced query, reporting, and creating the database.
For the desktop environment will be generated in JAVA code for various databases, the web environment is generated in PHP code for MySQL database.
In this way the work begins with a brief review of the literature about the concepts, standards and methodologies of programming languages and tools used in this.

Palavras chave: *Framework*, Java, Hibernate, PHP, Desenvolvimento WEB

SUMÁRIO

1 INTRODUÇÃO

Podemos dizer que nos dias de hoje existe um desafio constante na área de Engenharia de software, que é o de melhorar o processo de desenvolvimento. Cada dia mais as empresas de diversos ramos de atividades procuram soluções que atendam especificamente os seus requisitos em um tempo muito rápido.

Mesmo com a constante evolução de métodos, técnicas e ferramentas, a entrega de software em prazos e custos estabelecidos nem sempre é alcançada. A consequência dessa situação é o desenvolvimento de códigos desnecessários, falta de padronização nos métodos e nos atributos, e também repetição de tarefas com a mesma finalidade na programação. Em decorrência disso, aumentam-se os custos da produção diminuindo o lucro final, chegando até a ter prejuízo.

Uma das soluções apresentadas pelas empresas para resgatar o tempo perdido é o aumento de preço da prestação de serviço, feito isso, supostamente os cliente não ficarão contentes e seria um transtorno. Uma forma que as empresas podem trabalhar para conseguir driblar essa situação é usar ferramentas que as auxiliem automatizando e padronizando o projeto, juntamente com ferramentas geradoras de código. Um bom exemplo dessas ferramentas é o uso dos *frameworks*.

Um *framework*, segundo França(2000), é uma estrutura de classes inter-relacionadas, que constitui uma implementação inacabada, para um conjunto de aplicações de um domínio. Além de permitir a reutilização de um conjunto de classes, um *framework* também minimiza o esforço de desenvolvimento de novas aplicações, pois já contém a definição de arquitetura gerada a partir dele bem como, tem predefinido o fluxo de controle da aplicação.

Um gerador de código é uma ferramenta de software desenvolvida para a partir de um artefato de entrada flexível, gerar o código fonte de uma aplicação. Este código pode servir como base do desenvolvimento de uma aplicação ou simplesmente se tornar um protótipo, conforme LEAL(2005).

1.1 OBJETIVO

Este trabalho de Conclusão de Curso tem por objetivo o desenvolvimento de um *framework* denominado Java – Fácil. A utilização do referido *framework*, conforme mencionado anteriormente, pretende agilizar o processo de desenvolvimento de software, no tocante à criação de módulos CRUD *(Create, Read, Update e Delete)*, relatórios, consultas simples, consultas avançadas, menu principal, tabelas no banco de dados e fornecer classes que podem ser usadas por todo o projeto agilizando a construção da interface gráfica e também no acesso ao banco de dados. Os códigos serão gerados nas linguagens Java e PHP. Não se espera o desenvolvimento de um *framework* completo, mas um *framework* capaz de agilizar e padronizar o projeto.

1.2 JUSTIFICATIVA

O desenvolvimento do *framework* se baseia na facilidade de se ter os *feedbacks* dos clientes e boa parte do desenvolvimento do sistema inteiro em um curto período de tempo. Podendo criar simultaneamente no ambiente *WEB* e *DESKTOP* dando agilidade de fato na criação de código e padronização, sobrando tempo para as especificações do sistema.

1.3 MOTIVAÇÃO

O tempo de desenvolvimento de tarefas básicas, como montar telas, páginas e objetos para conexão com banco de dados é elevado, comparado com o tempo crítico do desenvolvimento, que é o processo de construção de regras de negócio.

A principal motivação para o desenvolvimento desse aplicativo deu-se pelo fato de ser uma tecnologia que ajuda nessas tarefas já mencionadas.

2 REVISÃO DA LITERATURA

2.1 LINGUAGEM DE PROGRAMAÇÃO

Uma linguagem de programação é um método padronizado para expressar instruções para um computador. É um conjunto de regras sintáticas e semânticas usadas para definir um programa de computador. Uma linguagem permite que um programador especifique precisamente sobre quais dados um computador vai atuar, como estes dados serão armazenados ou transmitidos e quais ações devem ser tomadas sob várias circunstâncias. (WIKIPEDIA, 2011)

Uma das principais metas das linguagens de programação é fazer com que o programador tenha mais interação com o que está sendo programado, sendo em uma linguagem mais fácil para entendimento humano. A linguagem de programação ao ser traduzida em linguagem computacional executa os comandos da mesma forma.

2.2 ORIENTAÇÃO A OBJETOS

Atualmente existem diversos tipos de linguagem de programação, e a que abordaremos neste trabalho é a programação orientada a objetos.

A programação orientada a objetos é um tipo de programação que destaca os objetos do mundo real, podemos dizer que é a abstração da regra de negócio para o mundo real, onde um objeto tem as suas características assim como os objetos do mundo real tem as suas.

Um bom exemplo de Orientação a objetos é um cadastro de cliente, se a programação não for orientada a objetos, os atributos do cadastro seriam passados um a um para a instrução que faz a inserção no banco de dados, se for orientado a objetos, ao invés de se passar um a um, passa-se um objeto com seus referidos atributos, dando mais entendimento ao que esta acontecendo.

Na programação orientada a objetos, implementa-se um conjunto de classes que definem os objetos presentes no sistema de software. Cada classe determina o comportamento (definidos nos métodos) e estados possíveis (atributos) de seus objetos, assim, como o relacionamento com outros objetos. Smalltalk, Perl, Python, Ruby, Php, C++, Java e C# são as linguagens de programação mais importantes com suporte à orientação a objetos. Neste trabalho abordaremos as linguagens Java e PHP.

2.3 JAVA

Java é uma linguagem de programação orientada a objetos, multiplataforma, e roda em qualquer sistema operacional aonde se tenha um interpretador instalado. Esse Interpretador denominado maquina virtual JAVA(JVM), é um programa que converte o código Java em comandos que o sistema operacional possa executar.

O Java utiliza um conceito diferente, ao invés de gerar um código binário diferente para cada plataforma, é gerado um binário que pode ser executado em qualquer plataforma, dentro de uma máquina virtual. Este código binário "universal" é chamado de *bytecode*.

Como Java é uma linguagem que não se prende a uma plataforma, ou a um sistema operacional, Java é utilizado em vários ambientes de desenvolvimento. Os mais conhecidos são o ambiente web, desktop e dispositivos celulares, estes sendo categorizados respectivamente por Java SE, EE, ME. Qualquer dispositivo que tenha uma JVM instalada, pode rodar programas escritos na linguagem Java. Isso facilita a vida do programador, pois pode reutilizar códigos de um ambiente para o outro sem se preocupar, cabendo essa preocupação aos desenvolvedores da linguagem JAVA, e dos criadores da JVM.

2.4 SWING

Mesmo que Java seja uma linguagem independente de sistemas ainda há a possibilidade de criar programas que dependam de determinado sistema operacional. Assim no começo da linguagem Java existia bibliotecas gráficas que rodavam somente em alguns sistemas operacionais.
Surgiu o *framework* Swing, que fornece componentes visuais de mais alto nível, possibilitando assim uma melhor compatibilidade entre os vários sistemas onde Java roda.

2.5 HTML

HTML significa Hyper Text Markup Language e é a linguagem de descrição de documentos usada na Web. A linguagem utiliza tags para definir os diferentes elementos, tais como texto, elementos multimédia, formulários, hiperlink, etc

Ao abrir uma página de site na internet, o usuário tem acesso direto com informações HTML, pois o navegador interpreta o documento, e o resultado final é mostrado na tela.

2.6 PHP

O PHP (*Personal Home Page*) é uma linguagem de programação dinâmica para produção de web sites. O PHP é processado no servidor, retornando para o cliente (pessoa que acessa o site) apenas HTML.

O PHP não é uma linguagem de programação COMPILADA como o C++ ou JAVA. É uma Linguagem de Programação Interpretada pelo servidor, pois não gera um executável, nem códigos de máquina.

Um script PHP pode conter ou não tags HTML, essas tags não são processadas pelo servidor, são simplesmente passadas ao solicitante. Normalmente utiliza-se HTML para fazer a estrutura e parte estática da página e o PHP para a parte lógica, que exige processamento.

2.7 JAVASCRIPT

Javascript é uma linguagem de programação com alguns recursos de orientação a objetos, é usado em sites para deixar a pagina HTML mais dinâmica, ou seja o javascript tem o poder de em tempo de execução criar mais tags HTML e exibir no navegador sem que a pagina tenha que ser atualizada.

2.8 CSS

Css (*Cascating Style Sheets*) é uma linguagem utilizada para adicionar estilos aos documentos web. Estilos que definem a apresentação dos conteúdos e a aparência das páginas. Em conjunto com o HTML tem o poder de mudar todo o estilo de apresentação de forma muito rápida e dinâmica. Pois o Css pode ficar separado do HTML, e o HTML usa os estilos criados no Css. Apenas mudar um estilo no Css, muda todos os componentes do HTML que estão utilizando aquele estilo.

2.9 BANCO DE DADOS

Com a necessidade das empresas guardarem grandes volumes de informações, de forma simples, rápida e confiável, que não houvesse a necessidade de se deslocar para acessar estas, que vieram os primeiros conceitos de armazenamento.

No início do desenvolvimento de software, guardavam essas informações em arquivos, onde estes podiam ser acessados pela rede. Mesmo assim ainda havia

grande dificuldade em alterar registros, estruturas ou simplesmente acessar alguns registros.

Com o objetivo de minimizar essa dificuldade surgiram os bancos de dados, estruturas com a capacidade de armazenar dados, controlar registros, controlar estrutura dos dados, sendo estes dados estruturados, que pode ser desde uma simples lista de compras a uma galeria de imagens ou a grande quantidade de informação da sua rede corporativa. Os computadores lidam muito bem com grandes quantidades de dados, o gerenciamento de banco de dados funciona então como a engrenagem central da computação, seja como utilitários independentes ou como partes de outras aplicações.

3 PROPOSTA DE DESENVOLVIMENTO DO *FRAMEWORK* JAVA FÁCIL

3.1 INTRODUÇÃO

Como já comentado anteriormente, o objetivo deste trabalho é o desenvolvimento de um *framework* voltado para o desenvolvimento de sistemas Desktop e Web que utilizam banco de dados como forma de persistência de informações. Para tanto, o *framework* provê uma arquitetura em três camadas, obedecendo ao padrão MVC de desenvolvimento de aplicações. O foco do *framework* está em oferecer um mecanismo automático, onde o desenvolvedor de sistemas não necessite codificar muitas linhas, que permita a persistência e recuperação de objetos em algum banco de dados. Neste contexto, o *framework* provê todas as funcionalidades para o desenvolvedor construir telas que permitam as operações rotineiras de um projeto. Este *framework* foi desenvolvido em Java, utilizando-se de outros *framework*s e ferramentas de código aberto. Nas próximas seções são abordados aspectos do desenvolvimento do *framework.*

3.2 FRAMEWORKS

3.2.1 DEFINIÇÕES

Várias definições sobre *framework* são descritas na literatura, mas segundo GAMMA (1995) "um *framework* é um conjunto de classes que cooperam entre si provendo assim um projeto reutilizável para um domínio específico de classes de sistema".

Um *framework* ou arcabouço é uma estrutura de suporte definida em que um outro projeto de software pode ser organizado e desenvolvido, quando se analisa o conceito no âmbito do desenvolvimento de software. Um *framework* pode incluir programas de suporte, bibliotecas de código, linguagens de script e outros softwares para ajudar a desenvolver e juntar diferentes componentes de um projeto de software.

Os *Frameworks* são projetados com o propósito de facilitar o desenvolvimento de software, habilitando projetistas e programadores a gastarem mais tempo detalhando as exigências de negócio do software do que com detalhes de baixo nível do sistema.
De acordo com GAMMA (1995) as características básicas devem ser respeitadas para que projetos de software sejam considerados um *framework*:

- Precisa ser reutilizável.
- Precisa facilitar o desenvolvimento de sistemas.
- Precisa possuir boa documentação.
- Precisa ser completo para o que se propõe.
- Precisa ser eficiente.

3.2.2 VANTAGENS E DESVANTAGENS

Utilizando *frameworks* a principal vantagem é a redução de custos, sendo que já existe uma estrutura definida e que o desenvolvimento pode concentrar-se em implementar as regras específicas do negócio em que o sistema deve atuar. Um *framework* ainda proporciona uma maior reutilização de códigos e a fatoração de problemas em aspectos comuns a várias aplicações, permite também obter sistemas com códigos menos frágeis e com menos defeitos.

3.2.3 TIPOS DE *FRAMEWORKS*

Classifica-se um *framework* de acordo com duas dimensões: como ele é utilizado e onde é utilizado. Quando tratamos de como um *framework* pode ser utilizado, analisamos o ponto de como introduzir as particularidades de uma aplicação. Neste sentido existem os *frameworks* Caixa-Branca, Caixa-Preta e Caixa-Cinza, de acordo com Maldonado(2001).

3.2.3.1 CAIXA-BRANCA

Os *frameworks* de Caixa Branca são baseados na especialização por herança e sobrescrita de métodos, com a disponibilidade de classes abstratas, que não podem ser instanciadas diretamente, podem ser herdadas e utilizar os recursos destas.

3.2.3.2 CAIXA-PRETA

São os *frameworks* focados na composição devendo utilizar as funcionalidades já presentes no *framework*, ou seja, neste tipo de *framework* as funcionalidades internas não podem ser vistas nem modificadas e devem utilizar as interfaces fornecidas pelo *framework*. Neste tipo as instanciações e composições feitas são o que determinam as particularidades da aplicação.

3.2.3.3 CAIXA CINZA

Os *frameworks* Caixa Cinza misturam os dois focos: herança e composição, ou seja, são *frameworks* baseados em herança (caixa branca) com algumas funcionalidades prontas.

3.2.4 QUALIDADES DE UM BOM *FRAMEWORK*

Para que um *framework* ofereça um bom suporte, é necessário flexibilidade para a ação que propõe atuar. Tendo um padrão em todo o projeto, não deixando o usuário confuso nas suas funções e especificações. Assim, devem-se utilizar os princípios de um projeto orientado a objetos, como o uso da herança para reutilização de

interface ao invés de reutilização de código e o uso do polimorfismo na definição das classes e métodos. As classes abstratas precisam estar no topo da hierarquia de classes, pois a finalidade destas classes é definir as interfaces a serem herdadas pelas classes concretas das aplicações.
A partir dessa situação é possível afirmar que o desenvolvimento de um *framework* é mais complexo que o desenvolvimento de uma aplicação específica. Para um bom funcionamento o *framework* deve possuir algumas qualidades como de acordo com a visão de (Silva) (2000);

Generalidade

Reflete a capacidade do framework em dar suporte a várias aplicações diferentes de um mesmo domínio, sendo flexível o suficiente para que as características de alterabilidade e extensibilidade possam ser aplicadas.

Alterabilidade

Reflete a capacidade do framework de alterar suas funcionalidades em função da necessidade de uma aplicação específica sem que estas alterações resultem em conseqüências imprevistas no conjunto de sua estrutura.

Extensibilidade

Reflete a capacidade do framework de ampliar sua funcionalidade sem conseqüências imprevistas no conjunto de sua estrutura. Ligada diretamente à manutenção do framework, permite que sua estrutura evolua por toda sua vida útil, pois à medida em que vai sendo utilizado, novos recursos vão sendo agregados para que o mesmo se ajuste as novas aplicações a que dá suporte.

Simplicidade

A estrutura geral do *framework* deve ser de fácil compreensão de forma que o desenvolvedor possa aprendê-lo em pouco tempo. Obviamente todos os detalhes do projeto não poderão ser aprendidos em poucos dias, porém o desenvolvedor deve estar apto para entender seu funcionamento em pouco tempo, deixando o

aprendizado de seus detalhes para o decorrer de sua utilização.

Clareza

Os aspectos comportamentais do *framework* devem estar encapsulados. Não há necessidade do desenvolvedor saber todos os detalhes de como o *framework* faz alguma coisa para que ele possa utilizá-lo.

Fronteiras

Um framework tem responsabilidades claras e sucintas, e deve acatá-las e nada mais. Todas as funcionalidades exteriores a fronteira do framework devem ser tratadas pelo desenvolvedor. Quando um framework ultrapassa esta fronteira, ele se torna complexo e provavelmente o desenvolvedor ao tentar utiliza-lo precisará implementar código adicional junto ao framework para conseguir o comportamento desejado.

3.3 FRAMEWORKS UTILIZADOS

3.3.1 HIBERNATE

Hibernate é um *framework* que permite ao desenvolvedor libertar-se de preocupações com a persistência de seus objetos. Com ele, pode-se fazer o mapeamento de objetos para bancos de dados relacionais, persistindo os objetos em tabelas de um banco de dados relacional, permitindo a convivência dos dois paradigmas (relacional e objetos) .

A relação entre o banco de dados e os objetos pode ser feita através de anotações ou com XML.

Oferece também métodos que podem ser utilizado para a criação das tabelas no banco de dados, com base nos objetos que são configurados como entidades no Hibernate.

As operações de criação, atualização, remoção e seleção de objetos são feitas através da API do Hibernate que, consultando seus arquivos de mapeamento e

configuração, envia os comandos corretos ao banco de dados configurado. Assim, a aplicação fica portável entre bancos de dados.

3.3.2 JASPER REPORTS

JasperReports é um poderoso *framework* open-source para geração de relatórios. Escrito em Java, essa biblioteca apresenta grande habilidade na organização e apresentação de conteúdo, permitindo a geração dinâmica de relatórios em diversos formatos, como PDF, HTML, XLS, CSV e XML, podendo ainda ser utilizada em qualquer aplicação Java, incluindo aplicações desktop, Web e distribuídas.

3.3.3 R& OS CLASS PDF

É um *framework* desenvolvido em PHP, com a intenção de gerar relatórios no formato pdf. Gera automaticamente o pdf em tempo de execução, e é muito fácil de se usar para a confecção de relatórios simples.

3.4 PADRÕES UTILIZADOS

3.4.1 MODEL VIEW CONTROL (MVC)

O padrão Model-View-Controller (BUSCHMANN, 1966) é um padrão de arquitetura que propõe a separação das classes de um sistema em três grupos: Model, View e Controller. O componente Model é o objeto da aplicação, o qual contém os seus principais dados e suas principais funcionalidades. O Model também provê funções para acessar seus dados. O objeto View recebe e apresenta informações para o usuário, mantendo a consistência com o respectivo, e o Controller define a maneira

com que a interface gráfica reage às ações de usuário (GAMMA, 1995; BUSCHMANN, 1996).

O objetivo principal do padrão MVC é separar dados e lógica de negócio (Model) da interface de usuário (View) e do fluxo da aplicação (Controller). Desta forma, os três componentes podem sofrer alterações sem impactar nos demais.

3.5 GERADOR DE CÓDIGO

Um gerador de código como o próprio nome diz, é uma ferramenta desenvolvida com o intuito de automatizar a criação de códigos de programação. Esse gerador de pode interagir com o usuário de diversas maneiras, uma delas é a de gerar o código de uma aplicação, sendo que esta estará sendo espelhada em um modelo de banco de dados. Neste trabalho, optou-se por utilizar a forma de interação da ferramenta com o usuário, onde o usuário digita as classes a serem geradas com seus determinados atributos.

As vantagens de um gerador de código (CODEGENERATION, 2006) são:

- qualidade do código: códigos gerados manualmente variam ao longo do projeto. É possível iniciar com alta qualidade e terminar em baixa ou vice-versa. Geradores de código padronizam a codificação minimizando pontos de falha, tendo em vista que *bugs* e não conformidades conhecidos são corrigidos e aplicados ao longo do sistema;

- consistência: códigos gerados seguem padrões de nomenclatura. Isso é benéfico para outros processos automáticos que usam este código (ex:Programação orientada a aspectos, AOP);

- produtividade: códigos automáticos são gerados em frações de segundos, comparados com o tempo que um programador levaria para gerar o mesmo código. A partir de outra perspectiva, os geradores possibilitam que o programador concentre-se em processos que exigem conhecimento no negócio, deixando um tempo maior para a programação que tem um tempo crítico;

3.6 FERRAMENTAS

Para o desenvolvimento do *framework* foram utilizadas as seguintes ferramentas:

3.6.1 ECLIPSE

O Eclipse é uma das ferramentas mais populares no mercado de desenvolvimento de software. Devido a sua extrema leveza, torna-se uma ferramenta ideal para trabalhar com a linguagem Java, já que a mesma emprega muitos recursos da máquina.

Destacam-se algumas funcionalidades no Eclipse, para o desenvolvimento em Java, que são:

- Auto-complete;
- Organização de *imports*;
- Auto-formatação de código;
- Mapeamento de referências;
- Compilação imediata;
- Geração de trechos de códigos;
- Incorporação de plugins: TO-DO, TASKS, etc.

O Eclipse é uma ferramenta de filosofia de código aberto *(Open-Source)*

3.6.2 IREPORT

Criar o design do relatório diretamente em XML pode ser uma tarefa custosa. Procuramos uma ferramenta que automatizasse esse processo. O iReport veio preencher essa lacuna, permitindo definir o design do relatório dentro de um ambiente gráfico, contento “todos” os recursos que a biblioteca Jasper oferece. É possível definir relatórios com designs modernos e complexos sem se quer escrever uma linha de código XML, que é gerado automaticamente. O ambiente ainda oferece atalhos para tarefas de compilação e visualização do relatório, permitindo a realização de testes, acelerando assim o processo de design.

É importante salientar que existem outras ferramentas com o mesmo objetivo que o iReport, mas que não são suficientemente maduras, no que diz respeito a facilidade de uso, e principalmente, no suporte as tags XML do JasperReports.

3.7 CATEGORIZAÇÃO DO *FRAMEWORK*

Dentre as classificações básicas de *framework*s podemos dizer que o objeto deste trabalho é um *framework* de domínio do tipo caixa-cinza, porque ele mescla características do caixa-branca com o caixa-preta. É caixa-branca no sentido em que customizações das ações básicas oferecidas pelo *framework* podem ser feitas através da herança como a classe Dao.

Da mesma maneira, pode-se dizer que é caixa-preta, pois permite a adição de novas regras de negócio e lógicas de acesso a dados, implementando as interfaces por ele disponibilizadas para cada camada. Entretanto, seu melhor aproveitamento se dá pela utilização das características de um *framework* caixa-branca, pois desta forma ele provê os serviços de forma completa, permitindo ao desenvolvedor customizar apenas algumas funcionalidades através de herança e sobrescrita de métodos

3.8 FUNCIONAMENTO GERAL DO *FRAMEWORK*

O *framework* é distribuído em um arquivo compactado, onde nesse arquivo existem as bibliotecas necessárias para a persistência no banco de dados(Hibernate), para a geração de relatórios(JasperReports), e para os bancos de dados Mysql, Hsqldb, Oracle. É distribuído também uma pasta chamada Css, que nela está contida os arquivos com funções em Javascript para manipulação de dados na web, juntamente com os arquivos .css que irão dar os modelos as interfaces das páginas web. A pasta pdf contém os arquivos do *framework* R& OS class PDF, que são os responsáveis por criar os pdfs no sistema web gerado.

O *Framework* é divido em dois arquivos:

gerador.jar – Distribui as classes necessárias para a persistência no banco de dados, e também para manipulação dos componentes swing.

geradorJF.jar – Responsável pela geração de código JAVA e PHP.

3.8.1 APLICAÇÃO MVC

O Framework gera o código dividindo as classes geradas por pacotes, para melhor organização do projeto. As classes referentes as transações com o banco de dados são criadas no pacote “dao”, as classes referentes ao modelo de persistência são criadas no pacote “objetos” e as Views são criadas no pacote “telas”.

3.8.2 JAVA - CAMADA VISÃO

A camada visão é criada dentro do pacote “telas” que é criado pelo próprio *framework*. As classes geradas são extendidas da classe JDialog, com exceção do menu principal que extende da classe JFrame.
No pacote “telas” ficam as classes geradas para “CRUD”, Consulta Simples, Consulta Avançada, Menu Principal e Login.
Essas classes usam as funções do pacote “dao” para controle dos objetos, sendo estes do pacote “objetos”.

3.8.3 JAVA - CAMADA CONTROLE

O Controle entre a tela e o banco de dados é realizado no pacote “dao”, onde todas as classes geradas para esse controle são extendidas da classe Dao que o *framework* fornece.

A classe Dao é abstrata, só pode ser usada quando extendida por outra. Fornece os principais métodos para manipulação dos registros no banco de dados. Porém todas as transações específicas devem ser feitas na classe que extende da classe Dao.

Para cada objeto é criada uma classe com o nome do objeto acrescentada de DAO no final.

3.8.4 JAVA - CAMADA MODELO

O modelo de dados que o *Framework* gera é de acordo com o preenchimento dos atributos na interface que o *framework* exibe.

Para toda classe criada o *framework* cria automaticamente dois atributos, que são “cancelado” e “causaCancel”, que são utilizados quando um registro é excluído, ficando salvo a causa do cancelamento do registro.

3.8.5 PHP - CAMADA VISÃO

A camada visão do PHP é separa pelas operações cadastrar, alterar, deletar e listar. As validações são realizadas por arquivos Javascript dentro da pasta css. O nome das classes é estabelecido de acordo com o nome dos objetos. Sendo que o menu principal é denominado “index.php”, onde são chamada as páginas de cadastros e relatórios.

3.8.6 PHP - CAMADA CONTROLE

A conexão do banco de dados fica em uma pasta chamada “conexao”. Os métodos de gerenciamento de banco de dados ficam na pasta “objetos”, onde o nome de cada arquivo é o nome do objeto adicionado “class.php”, contendo o objeto e os métodos.

3.8.7 PHP - CAMADA MODELO

O modelo de dados que o *Framework* gera está de acordo com o preenchimento dos atributos na interface que o *framework* exibe.

Para toda classe criada o *framework* cria automaticamente dois atributos, que são “cancelado” e “causaCancel”, que são usados quando um registro é excluído, gravando os motivos do cancelamento do registro.

3.9 JAVA - CLASSES UTILITÁRIAS

O *framework* contém classes de manipulação onde agiliza o processo no desenvolvimento de software. Entre estas classes estão:

- ManipulaValor – Contem métodos específicos para manipulação do tipo de dado Date. Conversões, soma de datas, diferenças, etc;
- ManipulaTime – Contem métodos específicos para manipulação do tipo de dado Double. Conversões, etc;
- MonetarioDocument – Classe usada para tipos de entrada de dados em campos como JTextField, onde o campo aceita apenas valores numéricos;
- InteiroDocument – Classe usada para tipos de entrada de dados em campos como JTextField, onde o campo aceita apenas valores inteiros;
- TextDocument – Classe usada para setar o tamanho da entrada de dados em campos como JTextField, onde o campo aceita o máximo de caracteres passado por parametro
- MudarFoco – Classe que padroniza os componentes de todo o sistema, métodos que manipulam as iterações entre os campos de um formulário;
- Relatorio – Classe que chama o relatório;

- VerificaOrdem – Classe que Exibe uma interface, onde se escolhe a ordem em que o relatório será exibido;
- JImagePanel – Classe que coloca um plano de fundo em um painel;
- Opcao – Classe que Exibe uma Interface, onde se escolhe a opção “Sim” ou “Não”;
- OpcaoP – Classe que Exibe uma interface, onde oferece um campo para entrada de dados;
- OpcaoS – Classe que Exibe uma interface, onde oferece um campo para entrada de dados de senhas;
- Dao – Classe que tem as principais operações para banco de dados, sendo uma classe genérica para objetos;
- HibernateUtil – Classe que abre as sessões que fazem as operações no banco de dados;
- DataGeneration – Classe que Cria as tabelas no banco de dados;

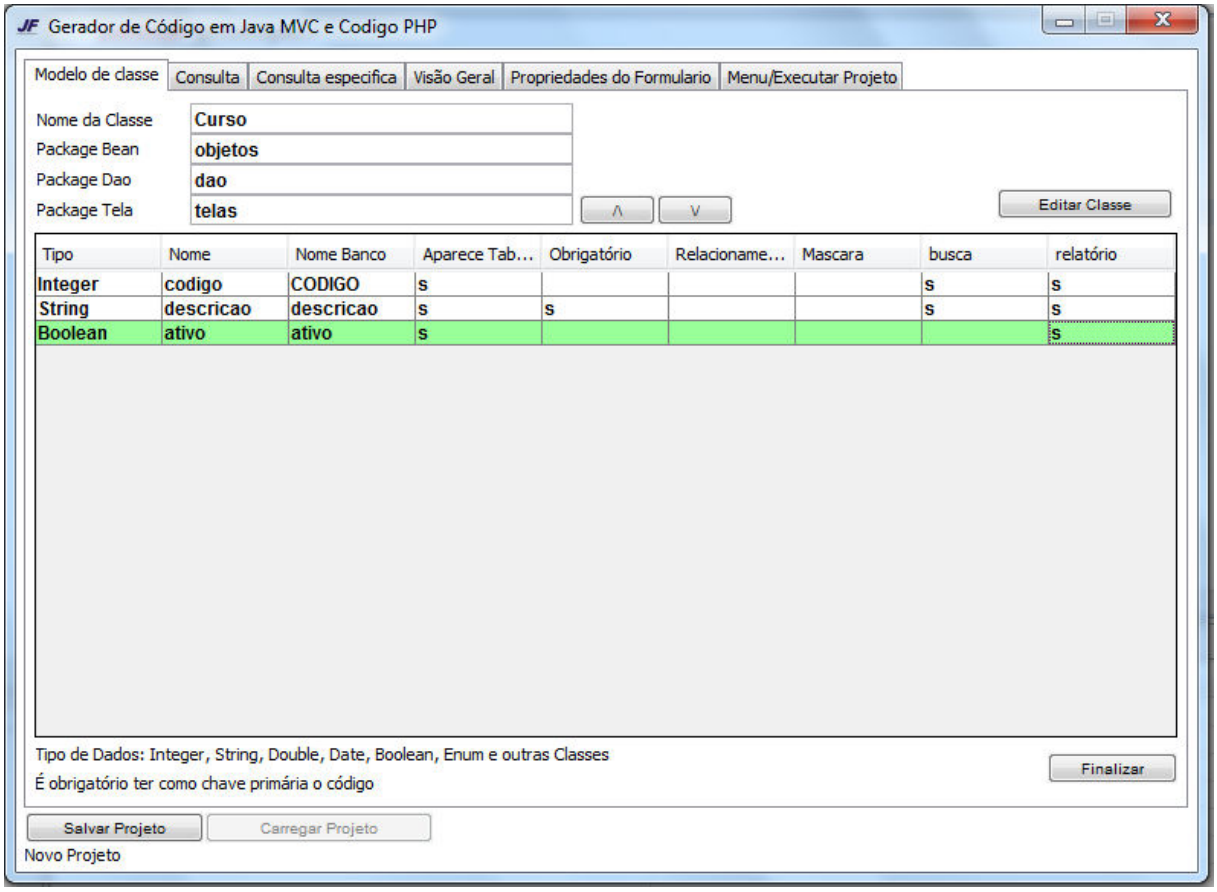
3.10 PHP – CLASSES UTILITÁRIAS

Na pasta “css” os arquivos disponíveis são:

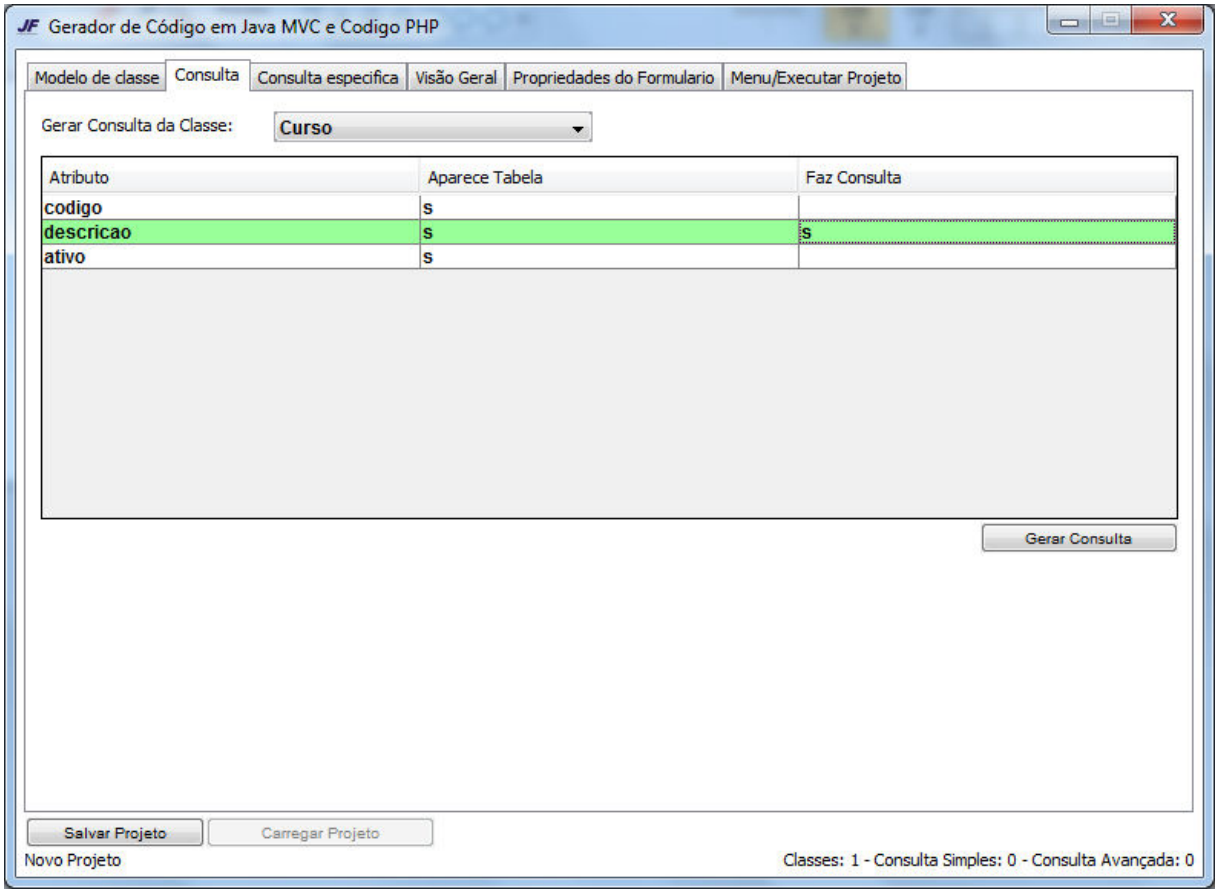
- SpryValidationTextarea – Contem métodos específicos para validação textarea.
- SpryValidationTextField – Contem métodos específicos para validação textfield.
- Gerenciador – Função para agilidade na hora de preencher campos de data.

Na pasta “pdf” estão as classes para a criação de relatórios.

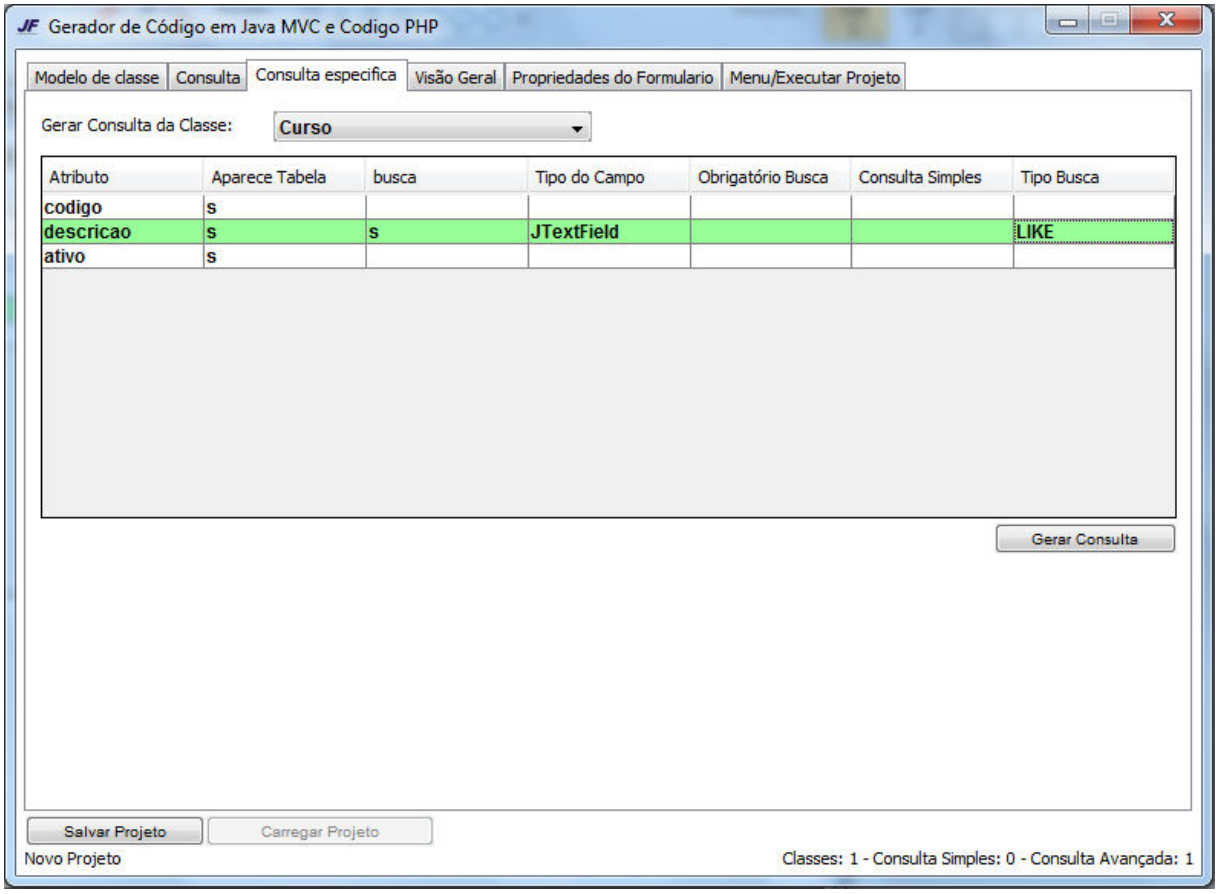
3.10 INTERFACES DO SISTEMA



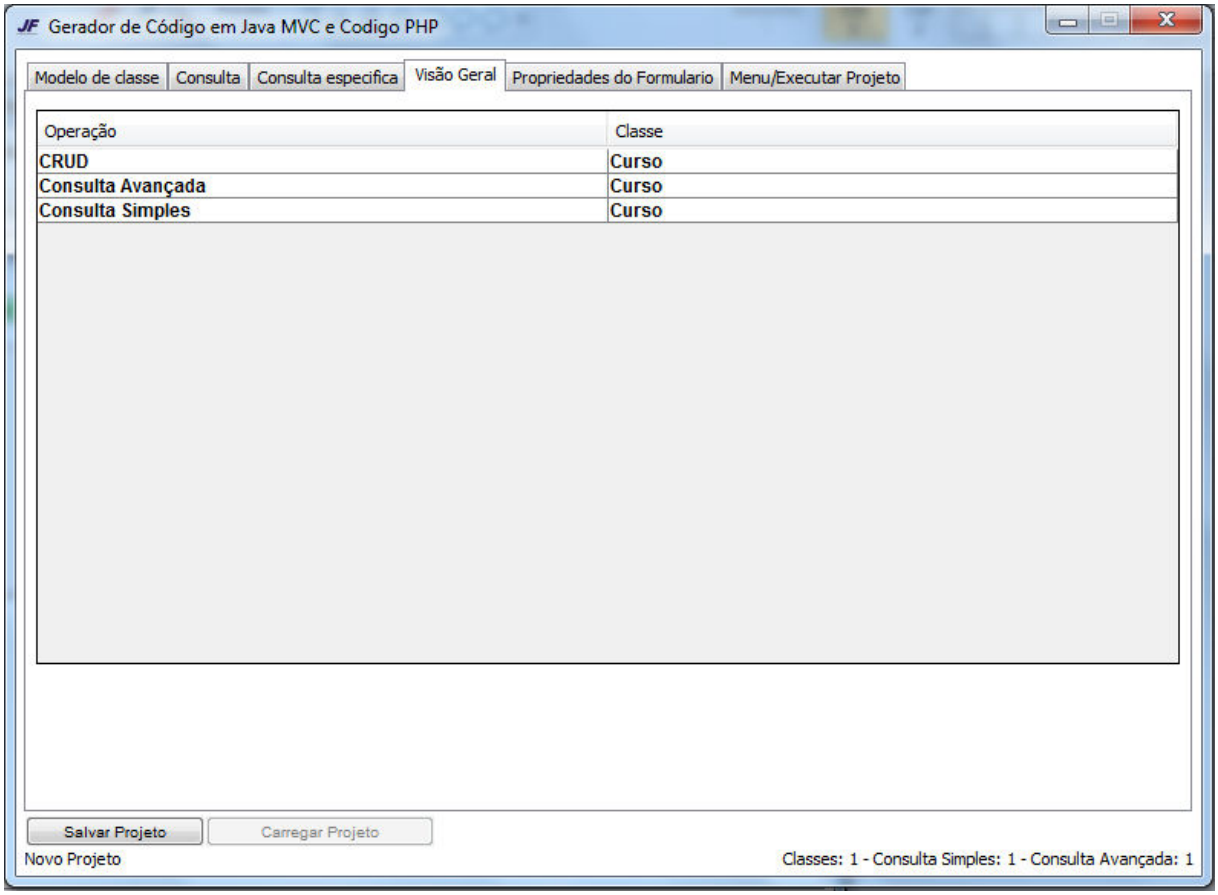
Esta é a interface para cadastrar classes no sistema com seus devidos atributos. A grid, que aparece no meio da tela, diz respeito aos atributos desta classe. As opções dessa funcionalidade são do tipo do atributo, propriedades do atributo, nome no banco de dados, opção de constar na tabela de consulta, obrigatoriedade, relacionamento, máscara, busca e relatório.



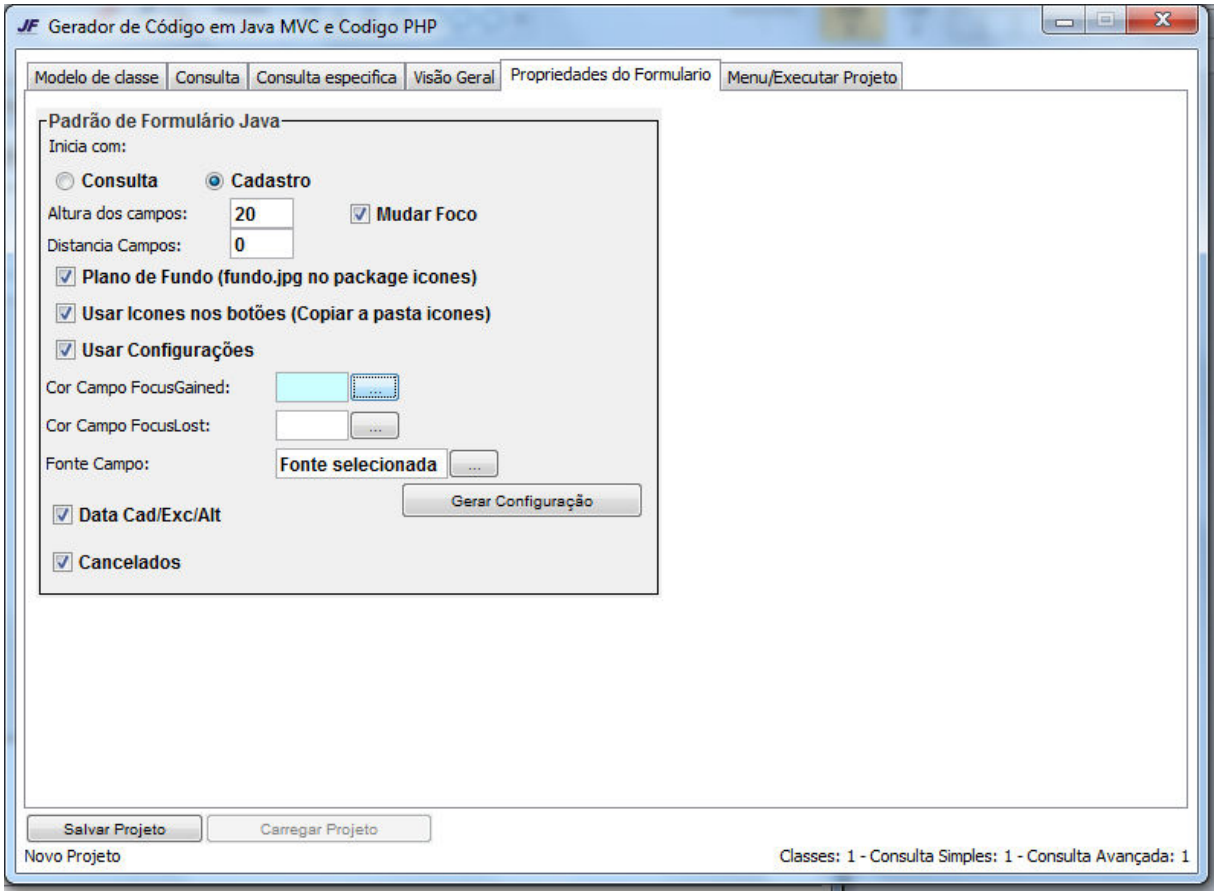
Esta Interface é a consulta simples das classes, é selecionado a classe a qual se deseja gerar a consulta simples. Ela apresenta as propriedades que constarão na tabela, e possibilita a opção de se fazer consulta ao atributo.



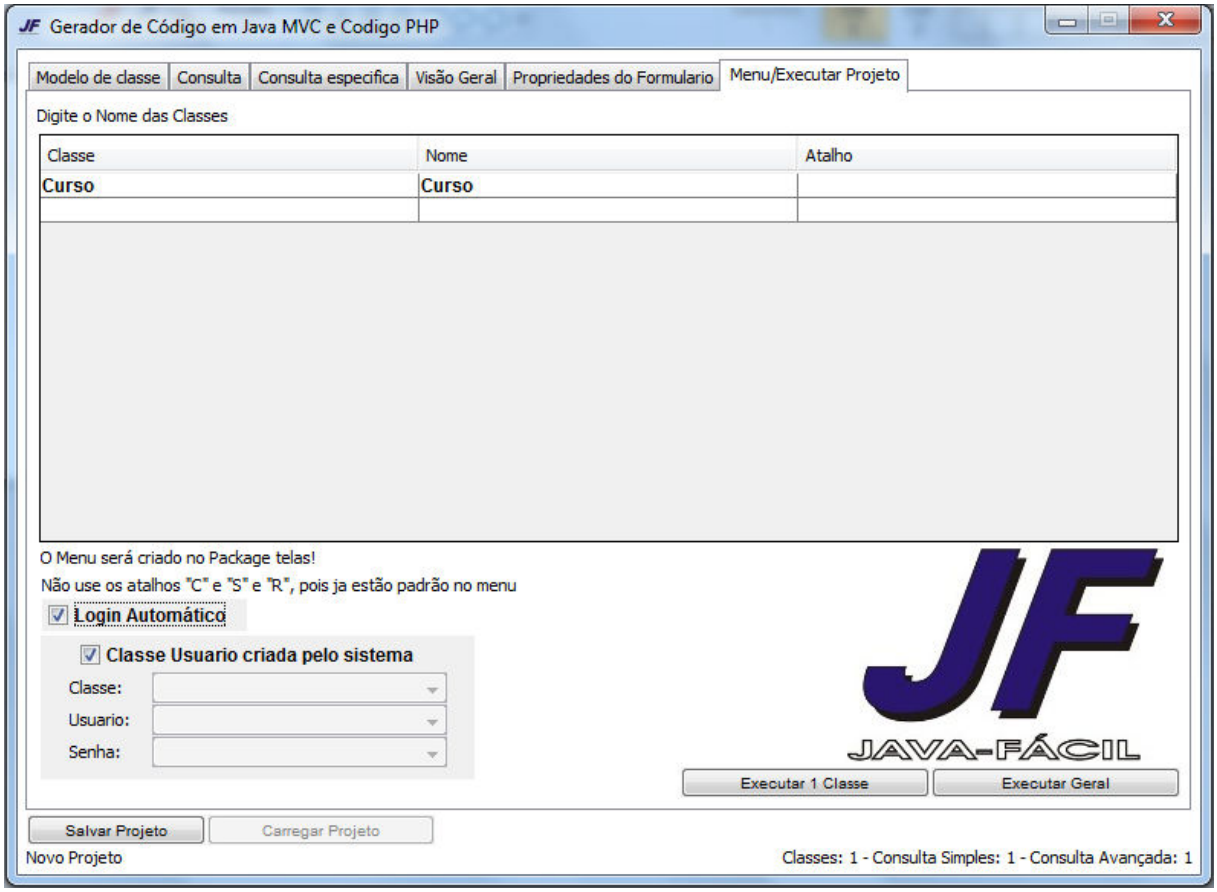
Esta tela é referente a consulta específica. Ao selecionar a classe que deseja criar uma consulta específica, são exibidos os atributos, logo após são preenchidos os campos das propriedades da consulta específica.



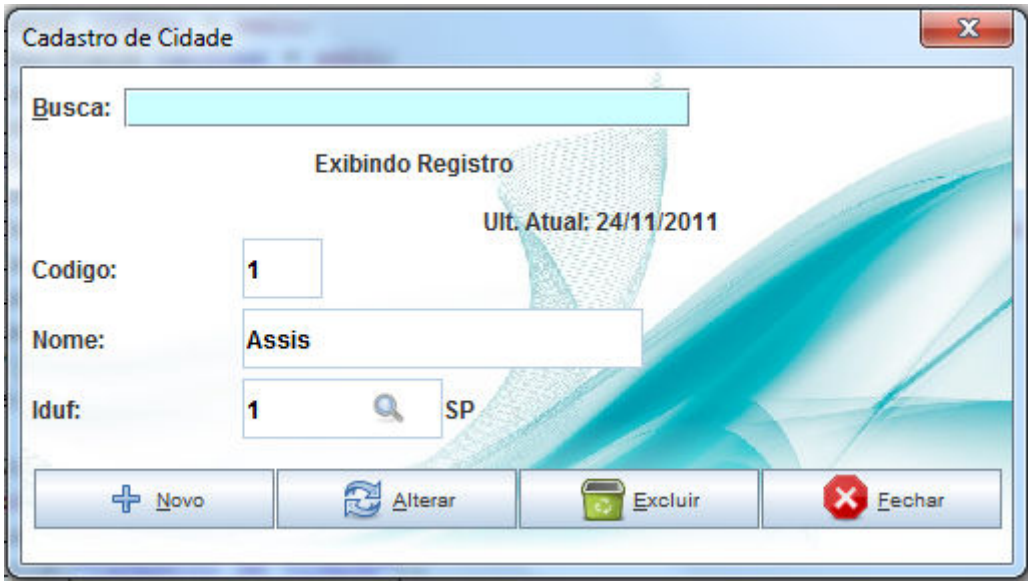
Esta Interface exibe a visão geral do projeto.



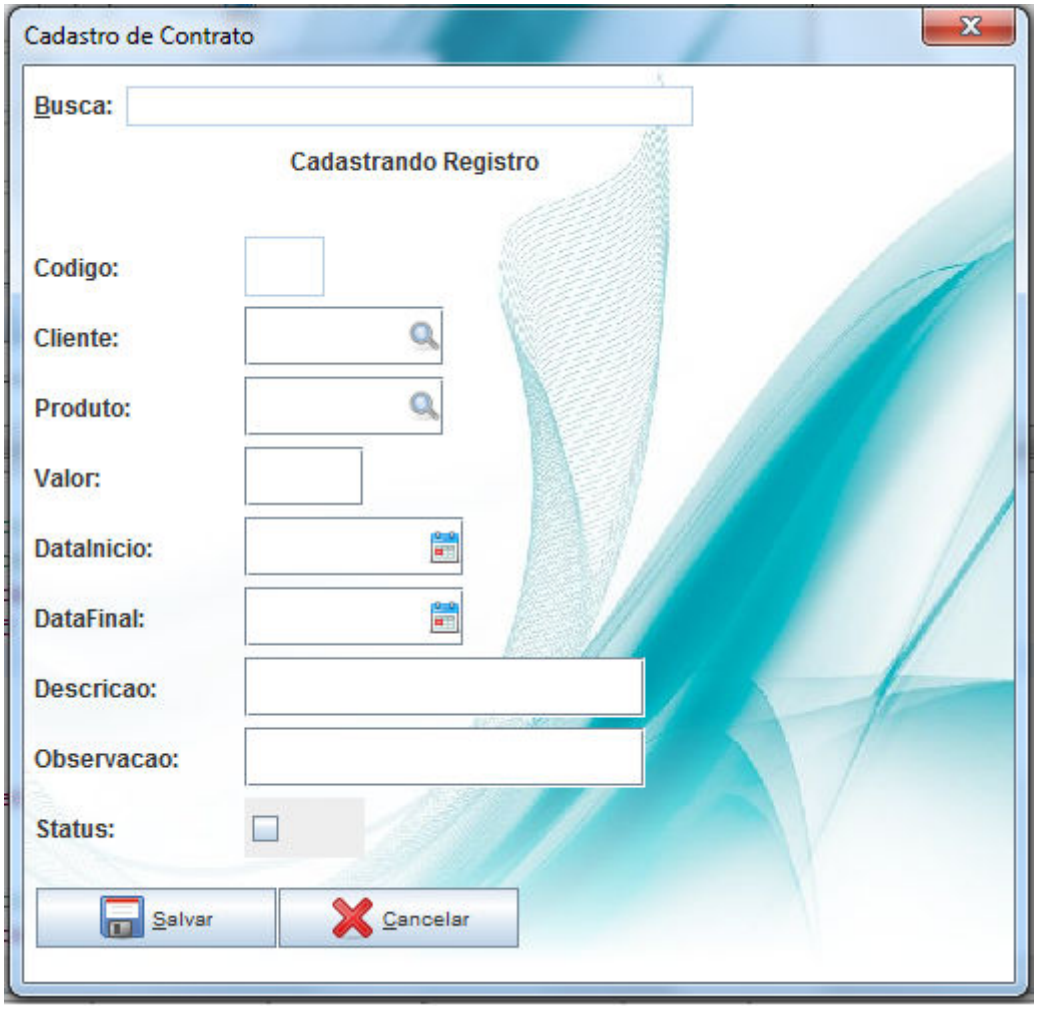
A partir dessa interface é possível informar as características sobre o formulário gerado: se o formulário começa em consulta ou cadastro, o tamanho dos campos em altura e largura, se muda o foco dos componentes ao apertar a tecla “Enter”, se haverá plano de fundo, se os botões serão compostos por ícones.



Esta interface exibe os Menus que serão gerados pelo menu principal, e apresenta a possibilidade de geração automática de login.



Esse formulário é um exemplo de cadastro gerado pelo framework.



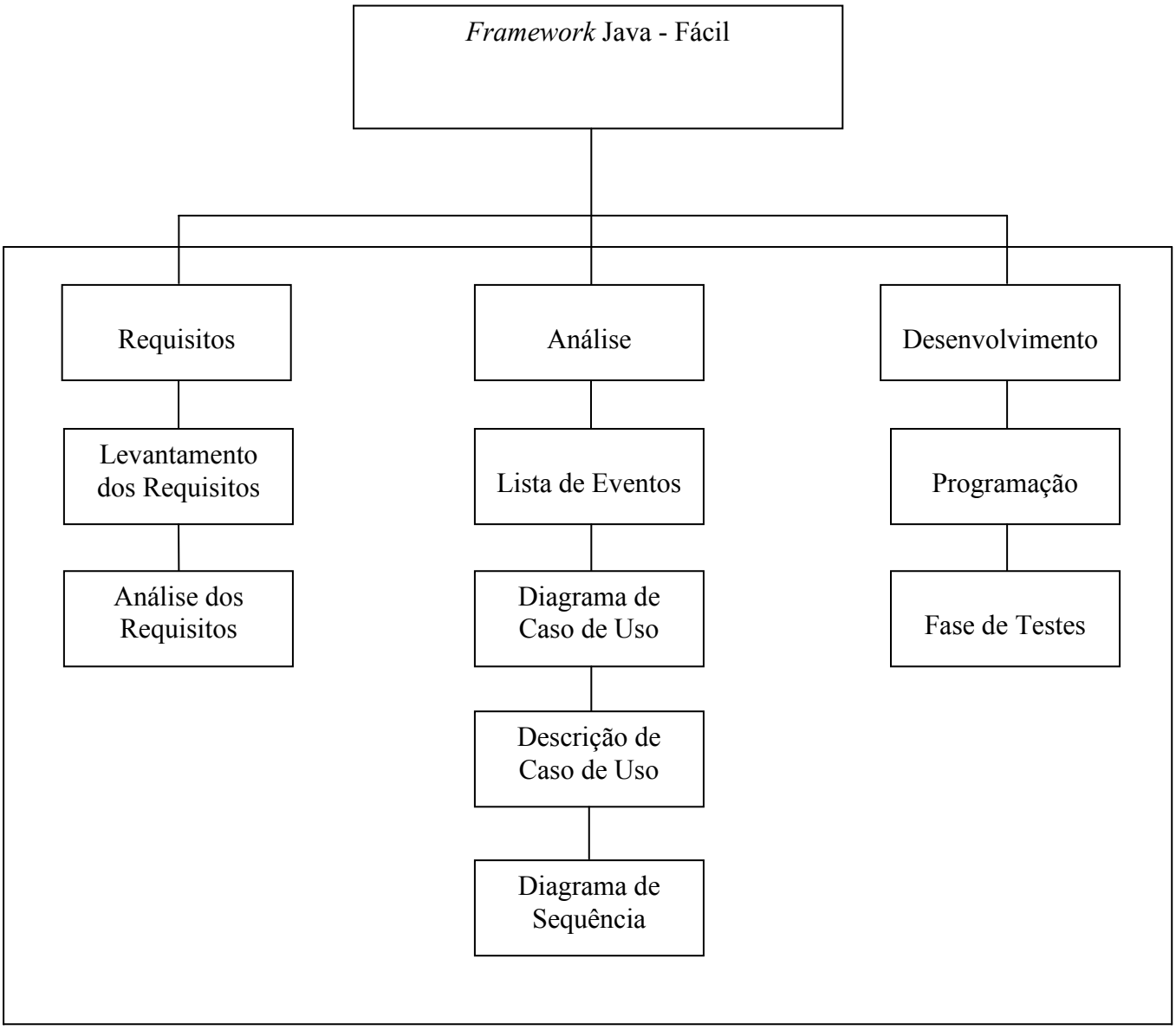
Outro exemplo de um formulário gerado pelo framework

Formulário PHP gerado pelo framework.

4 PLANEJAMENTO DO PROJETO

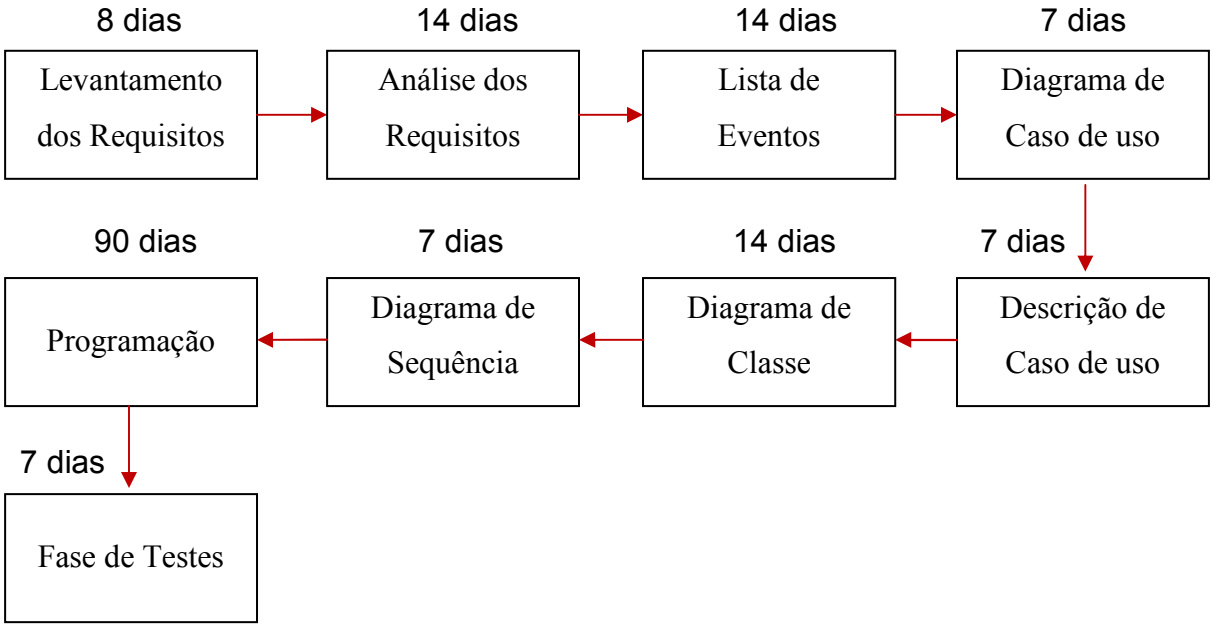
4.1 ESTRUTURAS ANALÍTICAS DO PROJETO (WBS)

Para o desenvolvimento do JAVA FÁCIL elaborou-se a seguinte Estrutura Analítica do Projeto:



4.2 SEQUENCIAMENTO DAS ATIVIDADES DEFINIDAS

O Sequenciamento de Atividades do projeto envolve estabelecer uma relação lógica das atividades a serem desenvolvidas ao longo do projeto JAVA FÁCIL:



4.3 CRONOGRAMA DE ATIVIDADES

Mês	Mar				Abril				Maio				Junho				Julho				Ag.				Set				Out				Nov			
Dia / Semana	05	12	19	26	02	09	16	23	07	14	21	28	04	11	18	25	02	09	16	23	06	13	20	27	03	10	17	24	01	08	15	22	05	12	19	26
Lev. Requisitos	x	x																																		
Análise requisito			x	x																																
Lista de Eventos					x	x																														
Diagrama UC							x																													
Descrição UC								X																												
Diagrama Classe									x																											
Diagrama Sequência										x																										
Programação											x	x	x	x	x	x	x	X	x	x	x	x														
Testes																							x													

4.4 ESTIMATIVA DE CUSTOS

Notebook: R$ 2.600,00

Depreciação de 2 anos: R$2.600,00/24 = R$108,00 Mensal ou 3,61Diário

Total de Custos do Sistema: R$ 2.600,00

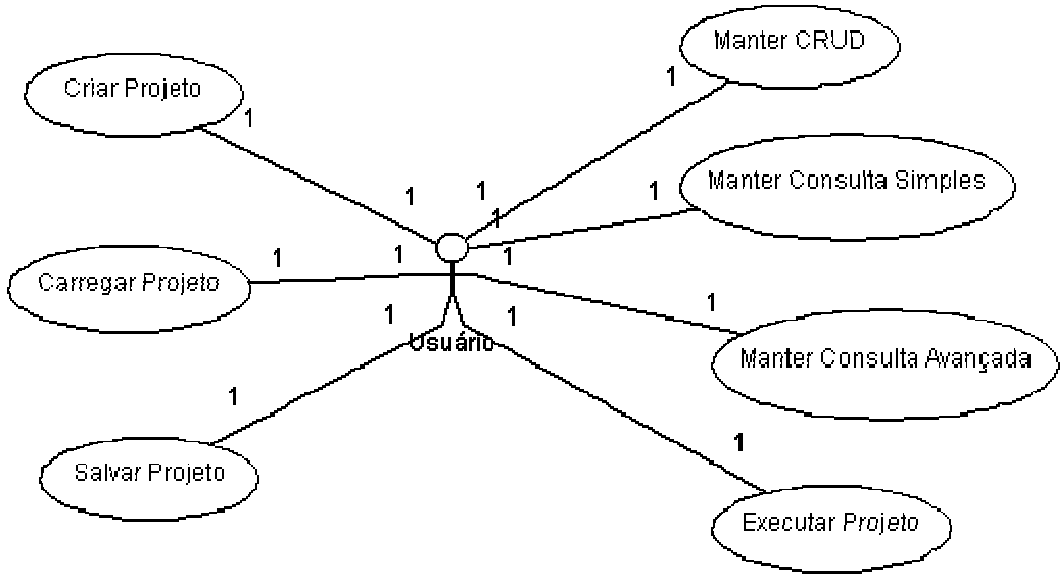
Orçamento do Projeto

Materiais	Valor em R$
Notebook	2.600,00
Total	2.600,00

4.5 LISTA DE EVENTOS

Nª	Nome	Objetivo	Caso de Uso
1	Usuário carrega um projeto	Seleção de um projeto já existente no computador	Carregar Projeto
2	Usuário Cria um CRUD	Tela de Cadastro completa	Criar CRUD
3	Usuário Cria consulta simples	Tela de consulta Simples	Criar consulta simples
4	Usuário Cria consulta avançada	Tela de consulta Avançada	Criar consulta avançada
5	Usuário Cria menu principal	Tela de menu principal	Criar Menu Principal
6	Manter CRUD	Usuário tem a opção de alterar uma classe ou excluir	Manter CRUD
7	Usuário salva o projeto	Salvar Projeto no computador	Salvar Projeto

4.6 CASO DE USO



4.7 DIAGRAMA DE CLASSE

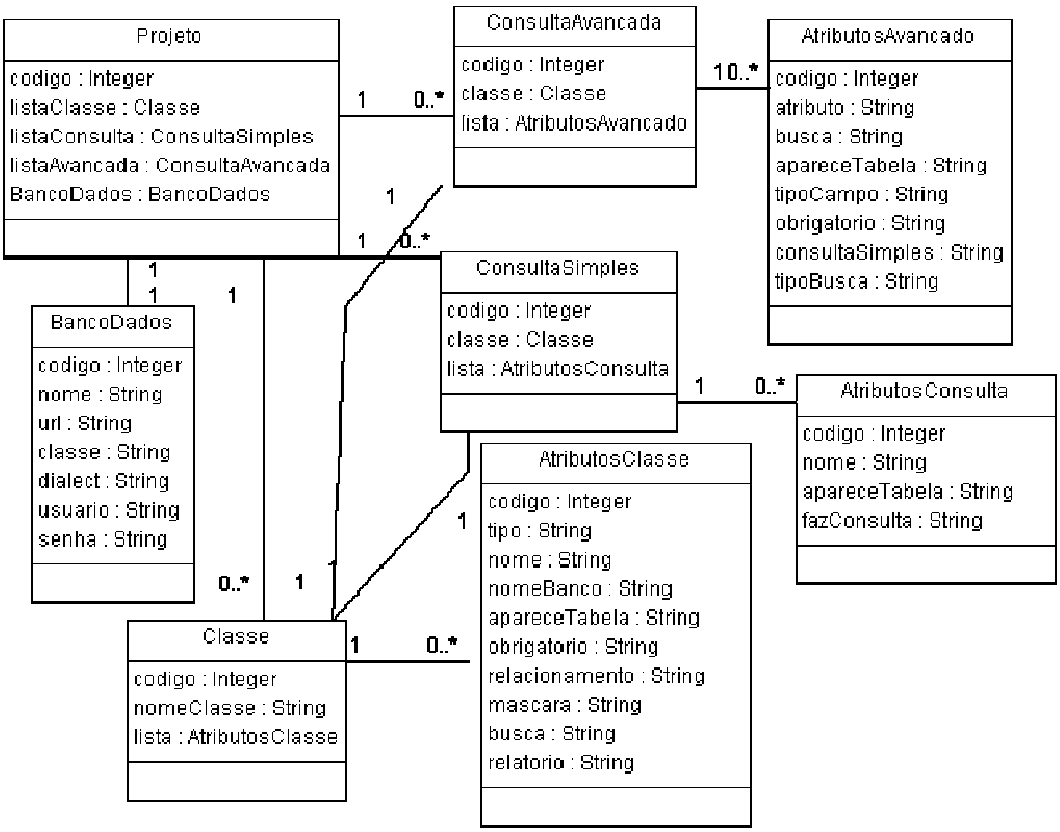


Figura 1 Diagrama de Classe

4.8 ESPECIFICAÇÃO DO CASO DE USO: Nº 1 - CARREGAR PROJETO

Finalidade/objetivo	Carregar um projeto salvo com a extensão jf;
Ator	Usuário;
Pré-condições	Existir um projeto salvo;
Evento Inicial	O Usuário da inicio clicando no botão “Carregar Projeto”;
Fluxo Principal	1 – Clicar no botão “Carregar Projeto”. [A1] 2 – O Sistema exibe a tela para encontrar o arquivo. 3 – Selecionar o Arquivo. 4 – O Sistema informa que o arquivo foi importado com sucesso;
Fluxo Alternativo	[A1] Novo Projeto 1 – Clicar no botão “Novo Projeto”. 2 – O sistema habilita os campos para criação do projeto;

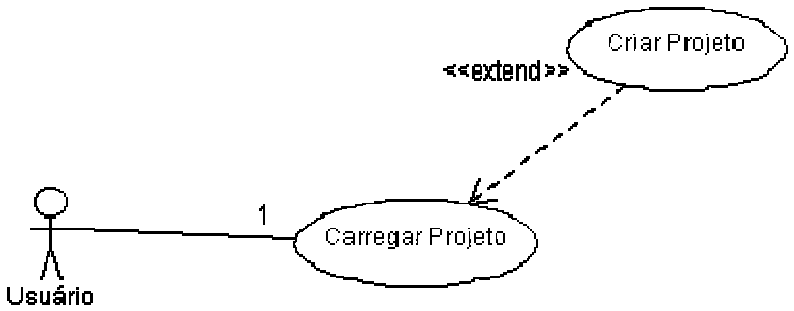


Figura 2 Caso de Uso Carregar Projeto

4.9 ESPECIFICAÇÃO DO CASO DE USO: Nº 2 - MANTER CRUD

Nome de Use Case	Manter CRUD
Descrição	A função deste caso de uso é criar um modelo de classe.
Ator	Usuário
Pré-Condições	Ter um projeto aberto.
Evento Inicial	O usuário seleciona a aba “Criar modelo de classe”.
Fluxo Principal	1 – Preencher os campos referentes ao novo modelo de Classe.[A1][A2] 2 – Clicar no botão “Finalizar”. [E1] 3 – O Sistema limpa os campos e informa que a Classe foi gerada com sucesso.
Fluxo Alternativo	[A1] Alterar Modelo de classe. 1 – O Usuário Escolhe uma classe para alterar. 2 – Altera os dados da classe. 3 – Clicar no botão “Salvar”. [A2] Excluir Modelo de Classe. 1 – O Usuário Escolhe uma classe para alterar. 2 – Clicar no botão “Excluir”.
Fluxo Exceção	[E1] Modelo sem atributos 1 – Sistema informa que o modelo está sem atributos.

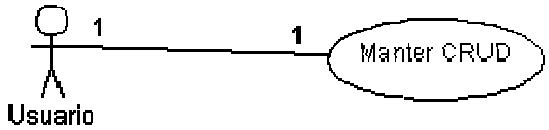


Figura 3 Caso de Uso Manter CRUD

4.10 DESCRIÇÃO DO CASO DE USO: Nº 3 - MANTER CONSULTA SIMPLES

Nome de Use Case	Manter consulta simples
Descrição	A função deste caso de uso é criar um modelo de consulta simples de determinada classe.
Ator	Usuário
Pré-Condições	Existir um modelo de classe no projeto.
Eventos Iniciais	O usuário seleciona a aba “criar consulta simples”.
Fluxo Principal	1 – Escolher a classe para gerar a consulta simples.[A1] [A2] 2 – Escolher os atributos para busca e para exibir na tela. 3 – Clicar no Botão “gerar”. [E1] 4 – O Sistema informa que foi gerado com Sucesso.
Fluxo Alternativo	[A1] Alterar Modelo de consulta. 1 – O Usuário Escolhe um modelo de consulta para alterar. 2 – Altera os dados da consulta. 3 – Clicar no botão “Salvar”. [A2] Excluir Modelo de consulta. 1 – O Usuário Escolhe um modelo de consulta para alterar. 2 – Clicar no botão “Excluir”.
Fluxo Exceção	[E1] Sem atributos selecionados para busca. 1 – Sistema informa que não foi selecionado nenhum atributo para busca.

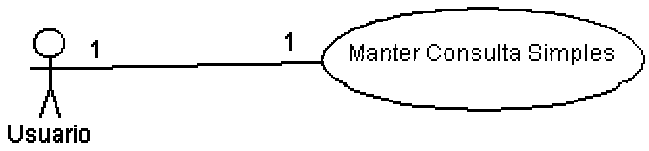


Figura 4 Caso de Uso Manter Consulta Simples

4.11 ESPECIFICAÇÃO DO CASO DE USO: Nº4 - MANTER CONSULTA AVANÇADA

Nome de Use Case	Manter consulta avançada
Descrição	A função deste caso de uso é gerar um modelo de consulta Avançada de determinada classe.
Ator	Usuário
Pré-Condições	Existir uma classe já finalizada no sistema.
Eventos Iniciais	O Usuário seleciona a aba “criar consulta específica”.
Fluxo Principal	1 – Escolher a classe para gerar a consulta avançada. [A1] [A2] 2 – Escolher os atributos para busca e para exibir na tela. 3 – Clicar no Botão “gerar”. [E1] 4 – O Sistema informa que foi gerado com Sucesso.
Fluxo Alternativo	[A1] Alterar Modelo de consulta avançada. 1 – O Usuário Escolhe um modelo para alterar. 2 – Altera os dados do modelo. 3 – Clicar no botão “Salvar”. [A2] Excluir Modelo de Classe. 1 – O Usuário Escolhe um modelo para alterar. 2 – Clicar no botão “Excluir”.
Fluxo Exceção	[E1] Sem atributos selecionados 1 – O Sistema informa que não foi selecionado atributos para a consulta.

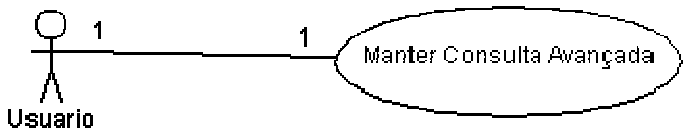


Figura 5 Caso de Uso Manter Consulta Avançada

4.12 ESPECIFICAÇÃO DO CASO DE USO: Nº5 – EXECUTAR PROJETO

Nome de Use Case	Executar Projeto
Descrição	A função deste caso de uso é gerar as classes objetos, dao, e tela, também criar os relatórios, criar o menu principal e gerar o banco de dados.
Ator	Usuário
Pré-Condições	Existir uma classe já finalizada no sistema.
Eventos Iniciais	O usuário seleciona a aba “Executar projeto”.
Fluxo Principal	1 – Clicar no Botão “Executar”. 2 – O Sistema pede para atualizar o Projeto. 3 – O Sistema pede para atualizar o Projeto. 4 – O Sistema informa que as classes foram geradas com sucesso. 5 – O Sistema pede a url do banco. [E1] 6 – O sistema informa que a operação foi realizada com sucesso.
Fluxo Exceção	[E1] – Falha ao criar o banco. O Sistema informa que não existe o banco indicado

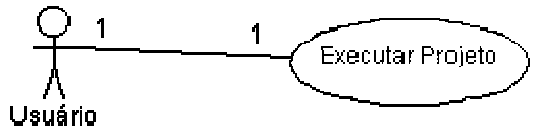


Figura 6 Caso de Uso Executar

4.13 ESPECIFICAÇÃO DO CASO DE USO: Nº6 - SALVAR PROJETO

Nome de Use Case	Salvar Projeto
Descrição	A função deste caso de uso é salvar o projeto com a extensão jf.
Ator	Usuário
Fluxo Principal	1 – Clicar no botão “Salvar Projeto”. 2 – O Sistema exibe a tela para localizar o local para salvar o arquivo. [A1] 4 – O Sistema informa que o arquivo foi salvo com sucesso!
Fluxo Alternativo	[A1] Arquivo já existente 1 - O sistema informa que foi salvo com sucesso.

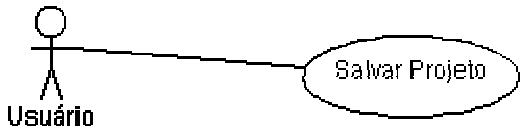


Figura 7 Caso de Uso Salvar

5 CONCLUSÃO

A procura de agilidade e padronização no desenvolvimento de software vem crescendo gradativamente. O framework desenvolvido nesse projeto visa atender essa procura para o ambiente Web na linguagem PHP e no ambiente desktop na linguagem JAVA.

O desenvolvimento deste trabalho tem contribuído expressivamente para o crescimento profissional do autor, uma vez que proporcionou conhecimentos específicos de tecnologias as quais ainda se não tinha conhecimento e frameworks já existentes no mercado.

REFERÊNCIAS

Grupo de Usuários Java, GUJ, http://www.guj.com.br. Acesso em 01/04/2011.

Ensino e Inovação, Caelum, http://www.caelum.com.br. Acesso em 03/04/2011.

Jboss, Hibernate – Relational Persistence for idiomatic java, http://www.hibernate.org/docs. Acesso em 27/03/2011.

França, L.; STAA, A. Geradores de Artefatos: Implementação e Instanciação de frameworks. Simpósio Brasileiro de Engenharia de Software, 15. SBC, 2001. PP 302-315.

LEAL, Marcelio D'Oliveira. CLASSGENERATOR: Um gerador de artefatos multiplataforma. Trabalho de Conclusão de Curso – Departamento de Informática – UFPA, 2005.

http://www.ros.co.nz/pdf/. Acesso em 02/06/2011.

MALDONADO, J. C.; BRAGA, R. T. V.; GERMANO, F. S. R.; MASIERO, P. C. Padrões e Frameworks de Software. Universidade de São Paulo

R.P. Silva. Suporte ao desenvolvimento e uso de frameworks e componentes. Tese de Doutorado, UFRS, 2000.

JOHNSON, Ralph. Documenting Frameworks Using Patterns. 1992. Disponível em: <http://citeseer.ist.psu.edu/johnson92documenting.html>. Acesso em: 1/08/2011.