UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

APLICAÇÃO DA TÉCNICA *ANALYTIC HIERARCHY PROCESS* (AHP) NA PRIORIZAÇÃO E SELEÇÃO DE PROJETOS

FRANCIELLE TONTINI ZIMERMANN

BLUMENAU
2012

2012/2-12

FRANCIELLE TONTINI ZIMERMANN

APLICAÇÃO DA TÉCNICA *ANALYTIC HIERARCHY PROCESS* (AHP) NA PRIORIZAÇÃO E SELEÇÃO DE PROJETOS

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciência da Computação — Bacharelado.

Prof. Everaldo Artur Grahl, Ms - Orientador

BLUMENAU
2012

2012/2-12

APLICAÇÃO DA TÉCNICA *ANALYTIC HIERARCHY PROCESS* (AHP) NA PRIORIZAÇÃO E SELEÇÃO DE PROJETOS

Por

FRANCIELLE TONTINI ZIMERMANN

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Everaldo Artur Grahl, Ms – Orientador, FURB
Membro:	Prof. Wilson Pedro Carli, Ms – FURB
Membro:	Prof. Jacques Robert Heckmann, Ms – FURB

Blumenau, 10 de Dezembro de 2012

Dedico este trabalho a todas as pessoas que acreditaram que eu fosse capaz para a realização do mesmo, especialmente a minha família e meu namorado.

AGRADECIMENTOS

A Deus, por agraciar-me com serenidade na busca e conclusão de mais uma etapa na minha vida.

À minha família, que sempre me apoiou e fez com que a conclusão deste trabalho fosse possível.

Ao meu namorado, que me incentivou e esteve sempre ao meu lado me ajudando e cobrando para realização do trabalho.

Aos meus amigos, pela compreensão em relação a minha ausência. para conclusão do trabalho.

Ao meu orientador, Everaldo Artur Grahl, por ter acreditado na conclusão deste trabalho e por ter se dedicado para possibilitar isso.

Os bons livros fazem “sacar” para fora o que a pessoa tem de melhor dentro dela.

Lina Sotis Francesco Moratti

RESUMO

A decisão sobre a seleção de um projeto e a priorização dos projetos selecionados são tarefas difíceis, pois cada projeto possui seus próprios benefícios, custos e riscos. Neste trabalho foi desenvolvida uma aplicação para apoiar a seleção e priorização de projetos utilizando a técnica *Analytic Hierarchy Process* (AHP). AHP é uma técnica multicritério e tem como objetivo estruturar uma decisão de maneira hierárquica, qualitativa e mensurável. A aplicação desenvolvida permite apontar qual projeto deve ser priorizado pela organização. Para isto, são definidos critérios por especialistas da organização e com base nestes critérios, os projetos são ordenados conforme a sua priorização. Essa ordenação é o resultado da aplicação da técnica AHP.

Palavras-chave: Técnica multicritério. AHP – *Analytic Hierarchy Process*. Seleção e priorização de projetos.

ABSTRACT

The decision on the selection of a project and the prioritization of the selected projects are difficult tasks, because each project has its own benefits, costs and risks. In this work an application was developed to support the selection and prioritization projects using the Analytic Hierarchy Process (AHP) technique. The AHP is a multi-criteria technique and has as aim to structure a decision in a hierarchical, qualitative and measurable way. The application allows to point out which project should be prioritized by the organization. For this, criteria are defined by specialists organization and based on these criteria, the projects are ordered according to their priority. This ordering is the result application of the AHP technique.

Keywords: Multi-criteria technique. AHP - Analytic Hierarchy Process. Selection and prioritization of projects.

LISTA DE ILUSTRAÇÕES

Figura 1– Critérios e grupos de critérios considerados na seleção de projetos 21
Quadro 1 - Cálculo do vetor de Eigen 22
Figura 2 - Hierarquia de critérios com as prioridades globais de cada um dos critérios 23
Figura 3 - Prioridades entre os critérios 27
Figura 4 - Priorização entre os projetos 28
Figura 5 - Peso entre os critérios por projeto 28
Quadro 2 - Requisitos funcionais 30
Quadro 3 - Requisitos não funcionais 31
Figura 6 - Diagrama de caso de uso 31
Quadro 4 - Caso de uso `UC01` 32
Quadro 5 - Caso de uso `UC01` 33
Quadro 6 - Caso de uso `UC03` 33
Quadro 7 - Caso de uso `UC04` 34
Quadro 8 - Caso de uso `UC05` 36
Quadro 9 - Caso de uso `UC06` 36
Quadro 10 - Caso de uso `UC07` 37
Quadro 11 - Caso de uso `UC08` 38
Quadro 12 - Caso de uso `UC09` 38
Quadro 13 - Caso de uso `UC10` 39
Figura 7 - Diagrama de atividades 40
Figura 8 - Pacotes da aplicação 42
Figura 9 - Pacotes `utils` 43
Figura 10 - Pacote `acompanhamento` 45
Figura 11 - Pacote `relatório` 46
Figura 12 - Pacote `grupocriterio` 47
Figura 13 - Pacote `projeto` 48
Figura 14 - Pacote `criterio` 49
Figura 15 - Diagrama de seqüência 50
Figura 16 - Menu da aplicação 51
Figura 17 - Cadastro de projeto 52

Quadro 14 - Adicionar um novo projeto – classe ProjetoGUI 52
Quadro 15 - Adicionar um novo projeto – classe `ProjetoDAO` 53
Quadro 16 - Classe que faz a conexão com o banco de dados 53
Figura 18 - Tela de pesquisa de projetos 54
Quadro 17 - Buscar lista dos projetos 55
Figura 19 - Acompanhamento de projetos 55
Figura 20 - Cadastro de grupo de critério 56
Quadro 18 - Evento do botão `Excluir` 56
Quadro 19 - Método excluir grupo de critério 57
Figura 21 - Cadastro de critério 57
Quadro 20 - Método do botão `Salvar` 58
Quadro 21 - Método alterar critério 58
Quadro 22 - Método mostrar tela 59
Figura 22 - Tela de boas-vindas 59
Quadro 23 - Buscar projetos 60
Quadro 24 - Adicionar projeto na tela de comparação de projetos 60
Figura 23 - Tela de seleção dos projetos 60
Quadro 25 - Método executado quando projeto é selecionado 61
Figura 24 - Tela de seleção dos grupos de critérios 61
Figura 25 - Tela de seleção dos critérios 62
Figura 26 - Tela para informar peso entre os grupos de critérios 62
Quadro 26 - Criação do *TableModel* para peso dos grupos de critérios 63
Quadro 27 - Criação da matriz para armazenar os pesos informados 63
Quadro 28 - Valores padrão da tela de peso 63
Figura 27 - Peso informado entre os grupos de critérios selecionados 64
Figura 28 - Peso informado entre os critérios selecionados 64
Quadro 29 - Cálculo do peso geral dos grupos de critérios 64
Quadro 30 - Cálculo do total da matriz 65
Quadro 31 - Cálculo da média da matriz 65
Quadro 32 - Cálculo do Vetor de Eigen 66
Figura 29 - Peso geral entre cada grupo e critério selecionado 67
Figura 30 - Informar peso entre os grupos de critérios 68
Quadro 33 - Exibir peso entre os projetos para cada grupo de critério 68
Quadro 34 - Soma dos pesos de todos os grupos de critério do mesmo projeto 71

Figura 31 - Seleção de relatório e exportar para XML 72
Figura 32 - Relatório dos projetos 72
Quadro 35 - Método para adicionar o peso geral do grupo de critério 73
Quadro 36 - Método `adicionarGrupo` 73
Quadro 37 - Montar os dados para a tabela que será utilizada no relatório 74
Quadro 38 - Método para gerar o arquivo PDF 74
Quadro 39 - Adicionar peso dos grupos e critérios 75
Figura 33 - Arquivo PDF com o relatório dos projetos 75
Quadro 40 - Lista com as informações da comparação de projetos 76
Quadro 41 - Método para exportar os projetos selecionados 76
Quadro 42 - Método para exportar os pesos entre os grupos de critérios 76
Quadro 43 - Método para exportar os pesos entre os projetos 77
Quadro 44 - Método para gerar o arquivo XML 77
Figura 34 - Exemplo do arquivo XML gerado 78
Figura 35 - Tela para gerar relatório de acompanhamento dos projetos 79
Quadro 45 - Método para buscar todos os acompanhamentos 79
Quadro 46 - Montagem da tabela com os acompanhamentos 80
Quadro 47 - Método para ler o arquivo XML 80
Quadro 47 - Método para ler o arquivo XML 80
Quadro 48 - Seleciona o grupo de critério na comparação de projetos 81
Quadro 49 - Informar peso conforme arquivo XML 81
Quadro 50 - Comparativo entre as características da aplicação e trabalhos correlatos 82

LISTA DE TABELAS

Tabela 1 - Matriz comparativa normalizada de cada grupo de critério 21
Tabela 2 - Cálculo do número principal de Eigen 22
Tabela 3 - Tabela de índices de consistência aleatória (RI) 23
Tabela 4 - Peso relativo de cada critério por projeto 24
Tabela 5 - Percentual de priorização do projeto mudança para novo escritório 24
Tabela 6 - Valor total da matriz entre os grupos de critérios 65
Tabela 7 - Cálculo da média das matrizes entre os grupos de critérios 66
Tabela 8 - Cálculo do vetor de Eigen para os grupos de critérios 66

Tabela 9 - Pesos entre os projetos para o grupo de critério Comprometimento 68
Tabela 10 - Pesos entre os projetos para o grupo de critério Financeiros 69
Tabela 11 - Pesos entre os projetos para o grupo de critério Outros critérios 69
Tabela 12 - Pesos entre os projetos para o critério Comprometimento da equipe 69
Tabela 13 - Peso geral entre os projetos para cada grupo de critério 70
Tabela 14 - Peso geral entre os critérios Comprometimento e Financeiros 70
Tabela 15 - Peso geral entre os critérios Outro critérios 70
Tabela 16 - Média entre os grupos de critério e critérios 71

LISTA DE SIGLAS

AHP – *Analytic Hierarchy Process*

CI – *Consistency Index*

CR - *Consistency Ratio*

RI – *Random Index*

TI – Tecnologia da Informação

TODIM - TOmada de Decisão Interativa Multicritério

XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO 15
1.1 OBJETIVOS DO TRABALHO 16
1.2 ESTRUTURA DO TRABALHO 16
2 FUNDAMENTAÇÃO TEÓRICA 17
2.1 GERENCIAMENTO DE PROJETOS 17
2.2 GESTÃO DE PORTFÓLIO 18
2.3 PRIORIZAÇÃO E SELEÇÃO DE PROJETOS 19
2.4 *ANALYTIC HIERARCHY PROCESS* 19
2.5 TRABALHOS CORRELATOS 25
2.5.1 Metodologia de apoio à decisão para priorização de projeto 25
2.5.2 Priorização de projetos, através de identificação e análise de critérios de seleção 26
2.5.3 Sistema computacional IPÊ 1.0 27
2.5.4 Sistema My choise, my decision 28
3 DESENVOLVIMENTO 30
3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO 30
3.2 ESPECIFICAÇÃO 31
3.2.1 Diagrama de casos de uso 31
3.2.1.1 Cadastrar projetos 32
3.2.1.2 Registrar acompanhamentos 32
3.2.1.3 Cadastrar grupos de critérios 33
3.2.1.4 Cadastrar critérios 34
3.2.1.5 Visualizar acompanhamentos de projetos 35
3.2.1.6 Importar XML 36
3.2.1.7 Realizar comparação 36
3.2.1.8 Listar projetos 37
3.2.1.9 Listar grupos de critérios e critérios 38
3.2.1.10 Exportar XML 38
3.2.2 Diagrama de atividades 39
3.2.3 Diagrama de classes 41
3.2.3.1 Pacote `utils` 42

3.2.3.2 Pacote `acompanhamento` 44
3.2.3.3 Pacote `relatorio` 45
3.2.3.4 Pacote `grupocriterio` 46
3.2.3.5 Pacote `projeto` 47
3.2.3.6 Pacote `criterio` 48
3.2.4 Diagrama de seqüência 49
3.3 IMPLEMENTAÇÃO 50
3.3.1 Técnicas e ferramentas utilizadas 50
3.3.2 Operacionalidade da implementação 51
3.3.2.1 Cadastrar projeto 51
3.3.2.2 Cadastrar grupo de critério 56
3.3.2.3 Cadastrar critério 57
3.3.2.4 Realizar comparação entre os projetos 59
3.3.2.4.1 Relatório de projetos 72
3.3.2.4.2 Relatório dos pesos dos grupos de critérios e critérios 75
3.3.2.4.3 Exportar para XML 75
3.3.2.5 Relatório de acompanhamento de projeto 78
3.3.2.6 Importar XML 80
3.4 RESULTADOS E DISCUSSÃO 82
4 CONCLUSÕES 84
4.1 EXTENSÕES 85

1 INTRODUÇÃO

A evolução tecnológica, a valorização do gerenciamento da qualidade e a globalização da economia fizeram com que todos os tipos de organizações aperfeiçoassem técnicas e estratégias para a melhoria de produtos, de processos e serviços, de modo a assegurar-lhes vantagens competitivas no mercado (SILVA; NASCIMENTO; BELDERRAIN, 2007). Sob esta perspectiva de competição e globalização, a gestão de portfólio de projetos é considerada uma atividade essencial para a sobrevivência de qualquer empresa. A gestão de portfólio remete a aplicação de recursos de forma a maximizar o retorno e minimizar os riscos nos projetos.

Saber qual projeto deve ser priorizado é um elemento impulsionador para o crescimento das empresas, porém, caso não seja realizada uma decisão correta, pode-se acarretar a perda de receitas, perda de mercado e em casos extremos, a falência da empresa (MODICA; ROQUE JUNIOR; BRAUN, 2011).

Com o intuito de auxiliar a seleção e priorização de projetos existem vários métodos que podem ser utilizados, dentre eles os métodos financeiros, os métodos de estratégia de negócio, os diagramas de bolhas, os *checklists*, os métodos de otimização e os métodos de apoio multicritério à decisão, como a teoria da utilidade, os métodos TOmada de Decisão Interativa Multicritério (TODIM) e o método de análise hierárquica - *Analytic Hierarchy Process* (AHP) (SILVA; NASCIMENTO; BELDERRAIN, 2007).

A decisão sobre a seleção de um projeto e a priorização dos projetos selecionados são tarefas difíceis, pois cada projeto possui seus próprios benefícios, custos e riscos. Estas características são na maioria dos casos raramente conhecidas com antecedência. A fim de tornar o processo decisório mais fácil, a organização pode utilizar metodologias de seleção e priorização de projetos, as quais necessitam ser avaliadas sob o aspecto do alinhamento com a estratégia traçada para a organização. Não se deve apenas utilizar os aspectos financeiros como fatores predominantes para a tomada de decisão, o que pode prejudicar os demais aspectos vitais para a implementação bem sucedida da estratégia da organização.

Apesar da importância de ser feita uma boa escolha na hora de decidir qual projeto deve ser priorizado, existem poucos *softwares* que auxiliam neste processo. Durante o uso da técnica AHP, por vezes é comum a utilização de planilhas, visando ilustrar os resultados ou mesmo facilitar a aplicação dos conceitos envolvidos.

Diante do exposto, neste trabalho foi abordado o desenvolvimento de uma aplicação para este fim. Foi utilizada a técnica AHP, um dos principais modelos matemáticos para apoio à tomada de decisão.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho foi desenvolver uma aplicação de apoio à priorização e seleção de projetos.

Os objetivos específicos do trabalho são:

a) disponibilizar na aplicação o algoritmo da técnica AHP para seleção de projetos;

b) apoiar as atividades de seleção e acompanhamento de projetos na área de Tecnologia da Informação (TI).

1.2 ESTRUTURA DO TRABALHO

O presente trabalho está estruturado em quatro capítulos. Nesse contexto, o segundo capítulo apresenta a fundamentação teórica necessária para o desenvolvimento do trabalho. Nele são expostos detalhes sobre o conceito de gerenciamento de projetos, partindo em seguida para o conceito de gestão de portfólio, explicando porque esta gestão é importante para o acompanhamento das atividades dos projetos. Em seguida é contextualizada a importancia da priorização da seleção de projetos para as organizações e por fim é explicada a técnica AHP (*Analytic Hierarchy Process*).

O capítulo também mostra características de alguns trabalhos correlatos. No terceiro capítulo é apresentado o desenvolvimento da aplicação de seleção e priorização de projetos, onde são apresentados os requisitos e a especificação da aplicação desenvolvida. Esta especificação compreende os diagramas de casos de uso, de atividades, de classes e de seqüência. O terceiro capítulo também demonstra a operacionalidade da aplicação e aborda aspectos relacionados à sua implementação, bem como os resultados obtidos. Finalizando, no quarto capítulo são apresentadas as conclusões e as sugestões para continuidade do estudo sobre a seleção e priorização de projetos.

2 FUNDAMENTAÇÃO TEÓRICA

Nas seções seguintes são detalhados os benefícios do gerenciamento de projetos e gestão de portfólio, explicação sobre priorização e seleção de projetos e apresentação do algoritmo utilizado na técnica *Analytic Hierarchy Process*. Por fim, são descritos os trabalhos correlatos.

2.1 GERENCIAMENTO DE PROJETOS

Segundo Sommerville (2003, p. 60), os *softwares* estão mais complexos, com cronogramas reduzidos e com mudanças freqüentes. Neste contexto, para os *softwares* com uma vida útil longa, o custo de manutenção é muito superior ao custo de construção.

Visando a redução dos custos são utilizadas técnicas e ferramentas de gerenciamento apropriadas que variam de acordo com o problema a ser resolvido, as restrições de desenvolvimento e os recursos disponíveis.

Segundo Project Management (2008), "projeto é um esforço temporário empreendido para alcançar um objetivo específico. Projetos são executados por pessoas, geralmente têm limitações de recursos e são planejados, executados e controlados.".

Para que um projeto seja executado, ele precisa ser gerenciado. Segundo Koontz e O'Donnel (1980, p. 319), "gerenciar consiste em executar atividades e tarefas que têm como propósito planejar e controlar atividades de outras pessoas para atingir objetivos que não podem ser alcançados caso as pessoas atuem por conta própria".

O gerenciamento de projetos cria um equilíbrio entre as demandas do escopo, tempo, custo, qualidade e bom relacionamento com o cliente. O sucesso no gerenciamento de projetos está relacionado ao alcance de alguns objetivos tais como a entrega dentro do prazo previsto, dentro do custo orçado, com nível de desempenho adequado, aceitação do cliente, atendimento de forma controlada às mudanças de escopo e respeito à cultura da organização (PROJECT MANAGEMENT, 2008).

Para satisfazer as necessidades de um projeto, pode-se elencar alguns itens principais, os quais são:

a) escopo, custo, tempo e qualidade;

b) partes interessadas com necessidades e expectativas diferenciadas;
c) requisitos identificados (necessidades) e requisitos não identificados (expectativas).

2.2 GESTÃO DE PORTFÓLIO

O gerenciamento de projetos é responsável pelas atividades de gerenciamento de um projeto em específico e a gestão de portfólio é responsável por envolver atividades relacionadas ao gerenciamento de um conjunto de projetos de uma empresa. Isso reúne as atividades de seleção dos projetos que farão parte do conjunto de projetos de uma empresa, a análise e a sua execução, para determinar se os projetos continuam viáveis e adequados em relação aos motivos pelos quais foram aprovados.

Pode-se entender um portfólio como sendo "(...) um conjunto de projetos, programas e outros trabalhos que são agrupados para facilitar o gerenciamento efetivo daquele trabalho para atender a objetivos estratégicos específicos" (PROJECT MANAGEMENT, 2008). Pode-se entender ainda que a gestão de portfólio refere-se ao "gerenciamento centralizado de um ou mais portfólios, que inclui identificar, priorizar, autorizar, gerenciar e controlar projetos, programas e outros trabalhos relacionados, para atingir objetivos estratégicos específicos" (PROJECT MANAGEMENT, 2008).

De forma geral, a gestão de portfólio de projetos é responsável por duas frentes, a de selecionar os projetos que devem ser executados e, uma vez em execução, acompanhar e avaliar se estes projetos continuam viáveis e aderentes aos critérios pelos quais foram aprovados.

Segundo Levine (2005, p. 24), o objetivo da etapa de seleção é conseguir encontrar os projetos que são adequados aos objetivos da empresa, alinhados com as suas estratégias e com as restrições de orçamentos e pessoas.

Já o objetivo da etapa de acompanhamento e avaliação é garantir que o projeto continua aderente e satisfazendo os objetivos pelo qual foi selecionado e iniciado. Esta etapa é responsável também por avaliar se o projeto continua sendo necessário frente às mudanças no ambiente que podem ocorrer durante a sua execução, tais como: novas tecnologias, requisitos legais, ofertas dos fornecedores, demanda dos clientes e mudanças na economia (MAIZLISH; HANDLER, 2005, p. 11).

2.3 PRIORIZAÇÃO E SELEÇÃO DE PROJETOS

Em grande parte das decisões organizacionais a intuição é responsável pela definição da escolha final. A intuição é o ato em que o decisor processa parte ou todas as informações que possui de maneira automática e rápida, sem considerar os detalhes envolvidos no problema. Decisões baseadas na intuição não levam em conta, de forma adequada, todas as informações disponíveis. Desta forma, os itens que deveriam influenciar na escolha não são os elementos determinantes do processo decisório (RUSSO; SCHOEMAKER, 1993, p. 9-31).

O ato de decidir é algo inerente ao ser humano. É algo em função de seu comportamento, de seus valores e de suas motivações. Sendo assim, a escolha do projeto que será priorizado possui relação direta com os valores e as motivações das pessoas que irão realizar esta decisão. Escolher é apostar em uma alternativa em função de preferências. Portanto, toda escolha implica em um sistema de preferências relativas às ações e às conseqüências destas ações.

Para que seja realizada uma efetiva seleção de projetos, devem-se estabelecer critérios claros e objetivos. Os critérios escolhidos pela empresa darão a fundamentação necessária para justificar a proposta de portfólio de projetos que será gerada após o passo da seleção. Além disso, a criação de critérios objetivos reduz a possibilidade de que decisões sejam tomadas com base em interesses políticos ou pessoais, visto que estabelece um processo lógico para a tomada de decisão.

Após realizar a seleção dos projetos será necessário priorizar estes projetos. Isso significa definir em que ordem os projetos selecionados devem ser executados. Esta priorização faz-se necessária, pois normalmente as empresas não possuem recursos suficientes para realizar simultaneamente todos os projetos selecionados. Da mesma forma que é feito na seleção, também é necessário estabelecer critérios para a tomada de decisão quanto à priorização dos projetos selecionados para o portfólio.

2.4 *ANALYTIC HIERARCHY PROCESS*

A programação multicritério por meio do *Analytic Hierarchy Process* (AHP) é uma técnica estruturada e tem como objetivo estruturar uma decisão de maneira hierárquica,

qualitativa e mensurável. Por conta destes múltiplos critérios, a técnica AHP proporciona vários benefícios.

O uso da técnica AHP facilita a estruturação do processo de tomada de decisão, melhorando a qualidade das decisões, minimizando os riscos e permitindo a integração e compartilhamento das informações entre os decisores.

Segundo Vargas (2010), a utilização do AHP é iniciada pela decomposição do problema em uma hierarquia de critérios que são mais facilmente analisáveis e comparáveis de modo independente. Após a criação desta hierarquia, os tomadores de decisão avaliam as alternativas por meio de comparação, de duas a duas, dentro de cada um dos critérios.

O AHP transforma as comparações, muitas vezes empíricas, em valores numéricos que podem ser processados e comparados. O peso que é definido para cada critério permite a avaliação de cada um dos elementos dentro da hierarquia definida. A capacidade de conversão de dados empíricos em modelos matemáticos é o principal diferencial do AHP com relação a outras técnicas comparativas.

Após a realização de todas as comparações e a atribuição de pesos relativos entre os critérios a serem avaliados, a probabilidade numérica de cada uma das alternativas é calculada. Esta probabilidade determina a probabilidade que a alternativa tem de atender a meta estabelecida. Quanto maior a probabilidade, mais aquela alternativa contribui para a meta final do portfólio.

Para realizar a construção do método AHP, são necessários os seguintes passos (VARGAS, 2010):

a) determinar os grupos e critérios que serão utilizados;

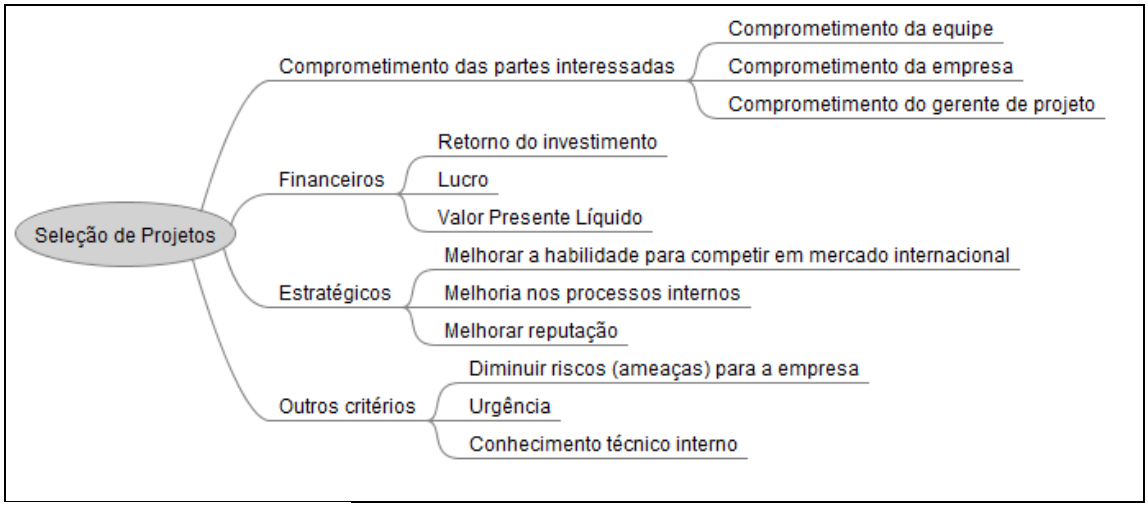
b) definir os pesos que cada critério possuirá;

c) calcular a contribuição de cada grupo de critérios, utilizando o vetor de Eigen;

d) verificar inconsistência nos dados;

e) priorizar os projetos conforme pesos dos critérios.

O primeiro passo na construção do AHP é a determinação dos critérios que serão utilizados. Cada empresa constrói seu próprio conjunto de critérios, que estão alinhados aos seus objetivos estratégicos organizacionais. Visando exemplificar os cálculos realizados pelo método AHP, serão considerados os critérios e grupos de critérios apresentados na Figura 1 (VARGAS, 2010).



Fonte: adaptado de Vargas (2010).

Figura 1– Critérios e grupos de critérios considerados na seleção de projetos

Após a escolha dos critérios que serão utilizados, os critérios precisam ser avaliados dois a dois, a fim de definir a prioridade relativa entre eles e seu peso relativo na meta global.

Na Tabela 1 são apresentados os pesos relativos entre os grupos de critérios, definidos pelos tomadores de decisão. Para que seja definido um peso relativo a cada grupo de critério é necessário normalizar a matriz comparativa (VARGAS, 2010). A normalização é feita pela divisão entre cada valor da matriz com o total de cada coluna, conforme pode ser vista também na Tabela 1.

Tabela 1 - Matriz comparativa normalizada de cada grupo de critério

	Comprometimento	Financeiros	Estratégicos	Outros critérios
Comprometimento	1	1/5	1/9	1
Financeiros	5	1	1	5
Estratégicos	9	1	1	5
Outros critérios	1	1/5	1/5	1
Total	**16,00**	**2,40**	**2,31**	**12,00**
Comprometimento	1/16 = 0,063	0,083	0,048	0,083
Financeiros	5/16 = 0,313	0,417	0,433	0,417
Estratégicos	9/16 = 0,563	0,417	0,433	0,417
Outros critérios	1/16 = 0,063	0,083	0,087	0,083

Fonte: Vargas (2010).

A determinação da contribuição de cada grupo de critério é calculada a partir do vetor de Eigen. O vetor de Eigen apresenta os pesos relativos entre os grupos de critérios e é obtido através da média aritmética dos valores de cada um dos critérios (VARGAS, 2010), conforme apresentado no Quadro 1.

Comprometimento	[0,063 + 0,083 + 0,048 + 0,083] / 4 = 0,0684 (6,84%)
Financeiros	[0,313 + 0,417 + 0,433 + 0,417] / 4 = 0,3927 (39,27%)
Estratégicos	[0,563 + 0,417 + 0,433 + 0,417] / 4 = 0,4604 (46,04%)
Outros critérios	[0,063 + 0,083, 0,087 + 0,083] / 4 = 0,0785 (7,85%)

Fonte: Vargas (2010).

Quadro 1 - Cálculo do vetor de Eigen

Os valores encontrados pelo vetor de Eigen determinam o peso de cada grupo de critério no resultado total da meta (VARGAS, 2010). Por exemplo, os critérios estratégicos têm um peso de 46,04% da meta total. Uma avaliação positiva neste fator contribui aproximadamente sete vezes mais do que uma avaliação positiva nos critérios de comprometimento (peso de 6,84%).

O passo seguinte é verificar se existe inconsistência nos dados. Esta verificação tem o objetivo de captar se os tomadores de decisão foram consistentes nas suas opiniões para a tomada de decisão.

O índice de inconsistência tem como base o número principal de Eigen. Ele é calculado através do somatório do produto de cada elemento do vetor de Eigen pelo total da respectiva coluna da matriz comparativa original (Tabela 1). A Tabela 2 apresenta o cálculo do número principal de Eigen (λMax).

Tabela 2 - Cálculo do número principal de Eigen

	Comprometimento	Financeiros	Estratégicos	Outros critérios
Vetor Eigen	0,0684	0,3927	0,4604	0,0785
Total	16,00	2,40	2,31	12,00
Valor principal de Eigen (λMax)	[(0,0684 x 16,00) + (0,3927 x 2,40) + (0,4604 x 2,31) + (0,0785 x 12,00)] = 4,04			

Fonte: Vargas (2010).

O cálculo do índice de consistência (SAATY, 2005, p. 346) é encontrado através da equação: CI=(λMax-n)/(n-1), em que CI é o índice de consistência e n é o número de critérios avaliados. Para o exemplo, o Índice de Consistência (CI – *Consistency Index*) é CI=(λ_Max-n)/(n-1)= (4.04-4) / (4-1) = 0,0143.

Para verificar se o valor encontrado do CI é adequado, Saaty (2005, p. 374) propôs o que foi chamado de Taxa de Consistência (CR - *Consistency Ratio*). Ela é determinada pela razão entre o valor do CI e o Índice de Consistência Aleatória (RI – *Random Index*). A matriz será considerada consistente se a razão for menor que 10%.

O valor de RI é fixo e tem como base o número de critérios avaliados, conforme a Tabela **3**.

Tabela 3 - Tabela de índices de consistência aleatória (RI)

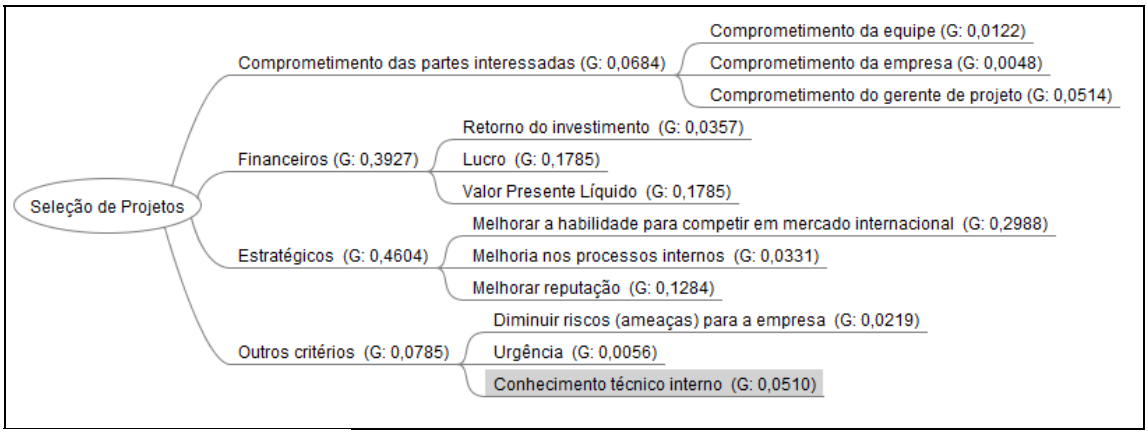
N	1	2	3	4	5	6	7	8	9	10
RI	0	0	0,58	0,9	1,12	1,24	1,32	1,41	1,45	1,49

Para o exemplo utilizado, a taxa de consistência para a matriz do grupo inicial de critérios é: CR=0,0143/0,9= 0,0159 = 1,59%. Como este valor é menor que 10%, a matriz pode ser considerada consistente.

Portanto, os valores apresentados no Quadro 1 podem ser considerados como o peso de cada grupo de critérios no resultado total da meta.

Da mesma maneira como foram feitos os cálculos para os grupos de critérios, torna-se necessário avaliar os pesos relativos dos critérios de cada grupo. Por exemplo, realizar os cálculos da técnica AHP entre os critérios de comprometimento do time, comprometimento da empresa e comprometimento do gerente de projeto.

Após a realização dos cálculos para todos os critérios, encontra-se a prioridade global de cada critério, conforme pode ser visto na Figura 2.



Fonte: adaptado de Vargas (2010).

Figura 2 - Hierarquia de critérios com as prioridades globais de cada um dos critérios

Com a árvore estruturada (Figura 2) e as prioridades dos critérios estabelecidas é possível determinar como cada um dos projetos candidatos comporta-se em relação aos critérios estabelecidos.

Da mesma forma que foi realizada para a priorização dos critérios, os projetos candidatos são confrontados dois a dois dentro de cada um dos critérios estabelecidos. Ao fim, o cruzamento entre todas as avaliações dos projetos em todos os critérios determinam a prioridade final de cada um dos projetos com relação à meta.

Para realizar a priorização entre os projetos Mudança para Novo Escritório e Abertura do Escritório na China, os tomadores de decisões comparam estes dois projetos entre os doze critérios definidos (Figura 2).

Primeiramente faz-se necessária a comparação dos dois projetos entre os critérios do grupo de comprometimento das partes interessadas.

Ao calcular-se todas as prioridades e os índices de inconsistências é possível determinar o peso relativo de cada um dos projetos em cada um dos critérios (Tabela 4).

Tabela 4 - Peso relativo de cada critério por projeto

	Comprometimento do time	Comprometimento da empresa	Comprometimento do gerente de projetos
Mudança para novo escritório	0,2968	0,0993	0,1586
Abertura do escritório na China	0,1613	0,4875	0,3444

Fonte: Vargas (2010).

O somatório do produto entre o peso de cada critério geral (Quadro 1) e o peso de cada critério no projeto (Tabela 4) determinam a prioridade que cada projeto possui. Por exemplo, considerando o projeto "mudança para novo escritório" e o grupo de critério "comprometimento", pode-se chegar ao valor apresentado na Tabela 5.

Tabela 5 - Percentual de priorização do projeto mudança para novo escritório

Critério	Peso do critério geral	Peso do critério no projeto	Produto
Comprometimento do time	0,0122	0,2968	0,0036
Comprometimento da empresa	0,0048	0,0993	0,0005
Comprometimento do gerente de projetos	0,0514	0,1586	0,0082
		Resultado	0,0123

Fonte: Vargas (2010).

Portanto, pode-se dizer que a priorização do projeto "mudança para novo escritório" dentre os itens do grupo de critério "comprometimento" é de 1,23%. Já para o projeto "abertura do escritório na China" é de 2,2%.

Portanto, entre os projetos "mudança para novo escritório" e "abertura do escritório na China" pode-se concluir que o projeto "abertura do escritório na China" é de maior prioridade, uma vez que seu peso em relação aos itens do grupo de critério "comprometimento" é de 2,2% contra 1,23% do projeto "mudança para novo escritório".

2.5 TRABALHOS CORRELATOS

É possível encontrar vários trabalhos que tenham relação com a utilização da técnica AHP para priorização de projetos. A seguir será mostrada a "Metodologia de apoio a decisão para priorização de projeto de tecnologia da informação" de Selma Foligne Crespio de Pinho (PINHO, 2006), a "Priorização de projetos, através de identificação e análise de critérios de seleção, relacionados aos objetivos estratégicos de negócio" de Eduardo Monteiro de Castro (CASTRO, 2010), o sistema computacional “IPÊ 1.0” de Helder Gomes Costa (COSTA, 2004) e o sistema “My choise, my decision” da empresa Init (INIT, 2007).

2.5.1 Metodologia de apoio à decisão para priorização de projeto

Este trabalho descreve uma metodologia para apoiar as organizações na priorização de projetos, uma vez que os recursos disponíveis não são suficientes para atender as demandas. Segundo Pinho (2006, p. 5), “o objetivo desta tese é apoiar a decisão quanto à priorização de projetos de tecnologia da informação a serem executados. Para a realização desta priorização, foram utilizadas técnicas de *data mining* (KDD) e análise multicritério à decisão”.

Neste trabalho foi utilizada a técnica de descoberta de Conhecimento em Bases de Dados (KDD - *Knowledge Discovery in Databases*) para obter as informações escondidas na base de dados da organização. O KDD tenta descobrir um padrão a partir de um conjunto de dados. Após ser detectado um padrão, é necessário possuir alguma medida de certeza ou relevância sobre os padrões descobertos. Os padrões gerados por esta técnica devem ser de fácil entendimento logo após a descoberta do padrão ou após algum processamento realizado por um especialista da organização. Por fim, são minerados os dados para que seja de fácil entendimento da organização.

Também é apresentada a técnica AHP para a priorização de projetos. Na utilização desta técnica, os critérios levantados pelos decisores são divididos em três níveis: superior, intermediário e inferior. Os critérios de um nível inferior são comparados aos níveis superiores e informados um peso para cada critério. Após a execução de todas as fases do método AHP, o vetor de prioridades resultante poderá ser utilizado para selecionar a alternativa de mais alta prioridade.

Como resultado da aplicação da metodologia, entre vinte e seis projetos que foram selecionados, apenas seis foram realizados pela organização. A partir dos resultados apresentados, o autor conclui que a metodologia proposta mostra-se um instrumento de auxílio à decisão que possibilita apresentar os resultados de forma priorizada. Segundo Pinho (2006, p. 112), "um percentual com maior prioridade, significa superioridade da alternativa sobre as outras, permitindo assim, avaliar com que intensidade uma alternativa é superior a outra, e até mesmo quais projetos não têm a sua execução recomendada.".

2.5.2 Priorização de projetos, através de identificação e análise de critérios de seleção

O objetivo do trabalho é auxiliar na priorização de projetos, considerando o portfólio de projetos e o orçamento disponível na organização. A priorização irá auxiliar na ordenação do projeto com base na sua relevância e prioridade para a empresa. Os critérios que foram considerados para a avaliação foram identificados por especialistas da empresa. Estes critérios são definidos com base no plano estratégico e dos objetivos estratégicos da organização (CASTRO, 2010, p. 7).

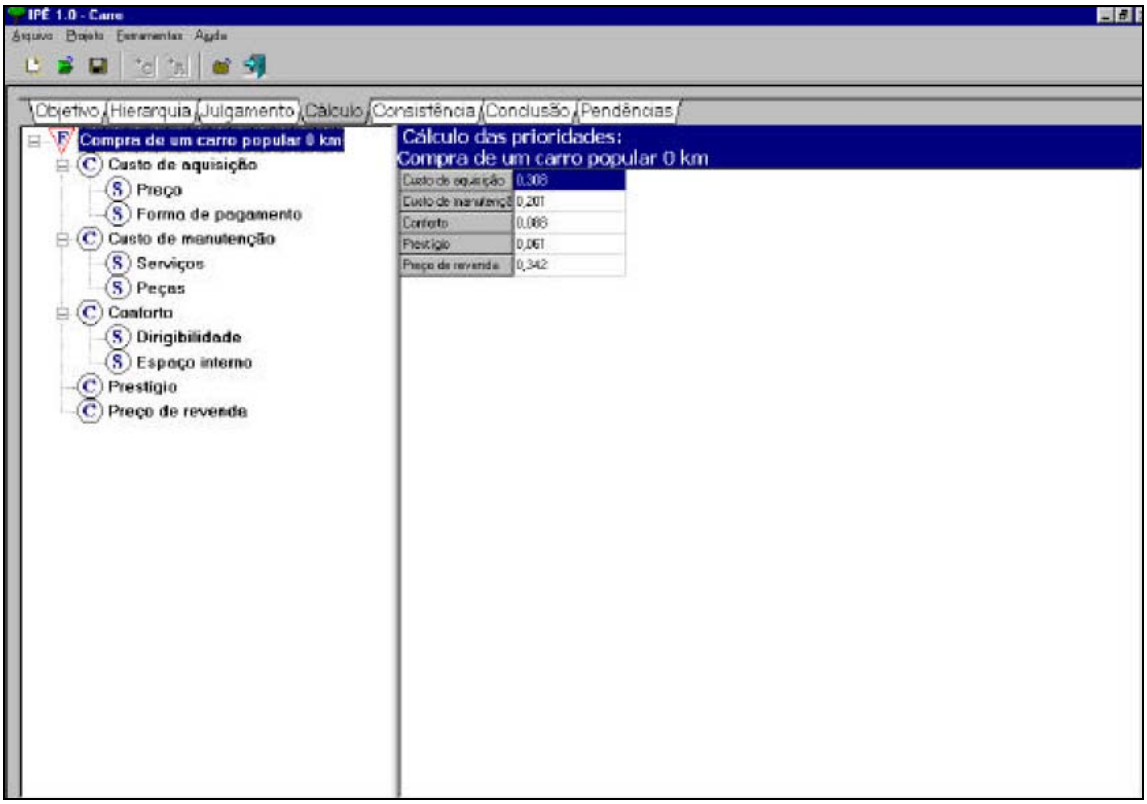
Neste trabalho também foi utilizada a técnica AHP. Para realizar a aplicação da técnica AHP a priorização foi divida em quatro fases. Na primeira fase são escolhidos os critérios, os cenários e projetos que serão avaliados. Esta escolha é feita pelos decisores da empresa. Na segunda fase é atribuído peso para cada um dos critérios elencados. Na terceira fase são atribuídos pesos aos critérios para refletir a relevância dos critérios nos cenários. Na última fase o decisor atribui notas de todos os projetos para cada um dos critérios identificados e determina os projetos que serão priorizados.

O trabalho foi validado na empresa Petróleo Brasileiro S/A. Foram avaliados quinze projetos e após a aplicação da técnica AHP foi apresentado um grafo ordenado pela priorização do projeto. A partir dos resultados apresentados o autor conclui que o esforço despendido *versus* o benefício alcançado indicam que os métodos em questão podem e devem ser utilizados. Tanto a revisão dos pesos e valores dos critérios, quanto a inclusão de novos cenários para serem analisados e comparados pode ser rapidamente executada. Ao final, Castro (2010, p. 75) afirma que "uma ferramenta de geração e simplificação de grafos é de extrema utilidade para complementar a planilha gerada para esta dissertação de mestrado.".

2.5.3 Sistema computacional IPÊ 1.0

A ferramenta proposta por Costa (2004) permite a aplicação da técnica AHP. Com ela, é possível cadastrar os critérios e grupos de critérios que serão utilizados pela técnica, definir o peso entre os critérios, realizar o cálculo do vetor de prioridades, avaliar o nível de consistência dos critérios, priorizar os critérios e priorizar os projetos com base na priorização dos critérios. As informações utilizadas na comparação são salvas em um arquivo localizado na máquina onde a aplicação está rodando.

A Figura 3 apresenta a tela com o cálculo das prioridades dos critérios.



Fonte: Costa (2004).

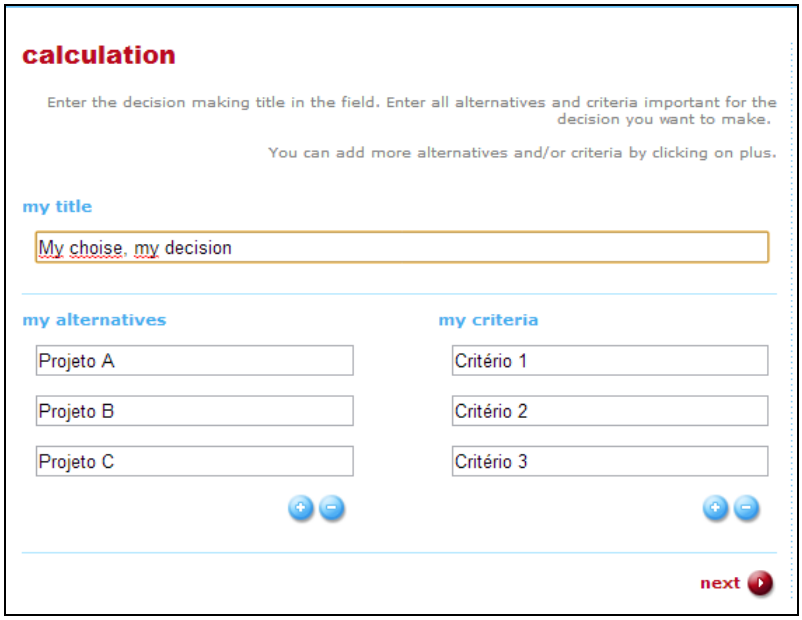
Figura 3 - Prioridades entre os critérios

Esta ferramenta foi testada em várias situações em um ambiente acadêmico. Nos testes foram construídas diferentes hierarquias, com vários critérios e sub-critérios. Nos testes finais, todos os resultados dos modelos executados pelo sistema IPÊ, coincidiram com os resultados obtidos manualmente. Isso indica que os resultados dos cálculos das prioridades e da análise de consistência estão de acordo com o método de análise hierárquica.

2.5.4 Sistema My choise, my decision

O sistema proposto pela empresa Init (2007) tem como objetivo realizar a priorização de projetos com base na aplicação da técnica AHP. No sistema são informados os projetos e critérios que serão considerados na priorização (Fonte: Init (2007).

Figura 4).

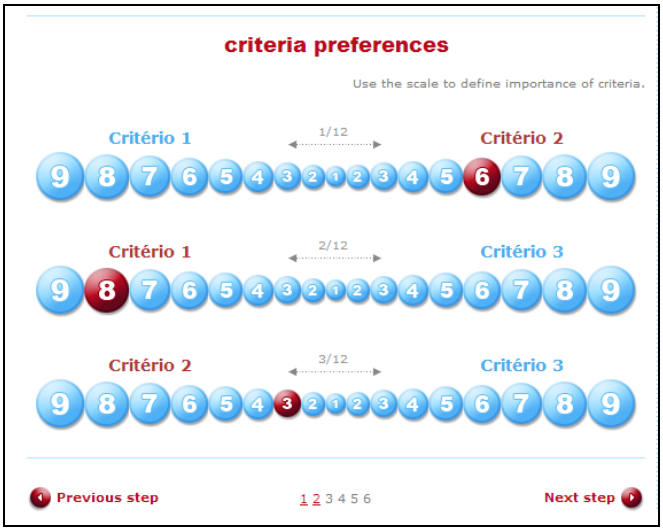


Fonte: Init (2007).

Figura 4 - Priorização entre os projetos

Em seguida são informados os pesos para cada critério entre os projetos informados (Fonte: Init (2007).

Figura 5).



Fonte: Init (2007).

Figura 5 - Peso entre os critérios por projeto

É realizado então o cálculo da priorização e apresentado o resultado final. O sistema é gratuito e pode ser acessado por qualquer pessoa pela internet. As informações utilizadas durante a comparação são salvas na conta do usuário conectado.

3 DESENVOLVIMENTO

Neste capítulo são apresentados os requisitos da aplicação, a especificação, a implementação e os resultados obtidos.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

Nesta seção são apresentados os requisitos funcionais (Quadro 2) e não funcionais (Quadro 3) da aplicação. Esses requisitos foram definidos com base nos estudos das ferramentas descritas nos trabalhos correlatos, levando em consideração também artigos e monografias estudadas neste trabalho.

No Quadro 2 é apresentada a associação entre os requisitos funcionais e os casos de uso, os quais serão descritos na seção 3.2.1.

REQUISITOS FUNCIONAIS	CASO DE USO
RF01: O sistema deve permitir ao usuário incluir, alterar ou excluir de projetos.	UC01
RF02: O sistema deve permitir ao usuário criar os critério e grupos de critérios.	UC03 e UC04
RF03: O sistema deve permitir ao usuário definir os pesos entre cada grupo, critério e projeto.	UC07
RF04: O sistema deve realizar a normalização da matriz comparativa entre os grupos, os critérios e os projetos.	UC07
RF05: O sistema deve calcular a contribuição de cada grupo, critério e projeto utilizando o vetor de Eigen.	UC07
RF06: O sistema deve calcular o número principal de Eigen para cada grupo, critério e projeto selecionado.	UC07
RF07: O sistema deve calcular a taxa de consistência da matriz de grupos, critérios e projetos.	UC07
RF08: O sistema deve permitir ao usuário gerar um relatório com a lista dos projetos ordenados por sua priorização.	UC08
RF09: O sistema deve permitir ao usuário gerar um relatório com a lista dos critérios ordenados por seu peso.	UC09
RF10: O sistema deve permitir ao usuário registrar o histórico de atividades dos projetos para facilitar o respectivo acompanhamento.	UC02
RF11: O sistema deve permitir ao usuário exportar a comparação dos projetos para um arquivo XML	UC10
RF12: O sistema deve permitir ao usuário importar a comparação dos projetos para um arquivo XML	UC06
RF13: O sistema deve permitir ao usuário visualizar o historio de acompanhamentos dos projetos	UC05

Quadro 2 - Requisitos funcionais

REQUISITOS NÃO FUNCIONAIS
RNF01: Ser implementado utilizando a linguagem de programação Java, versão 1.7.0.
RNF02: Utilizar o ambiente Eclipse, versão Indigo Service Release 1, para o desenvolvimento.
RNF03:Utilizar o banco de dados Oracle 10g.

Quadro 3 - Requisitos não funcionais

3.2 ESPECIFICAÇÃO

A aplicação foi especificada através da ferramenta *Enterprise Architect* (EA), utilizando os conceitos de orientação a objetos e baseando-se nos diagramas da UML, gerando como produtos os diagramas de caso de uso, de atividades, de classes e de seqüência apresentados nas seções seguintes.

3.2.1 Diagrama de casos de uso

A aplicação construída possui dez casos de uso (Figura 6). Todos os casos de uso são executados pelo ator `Usuário`, que representa a pessoa que faz uso da aplicação.

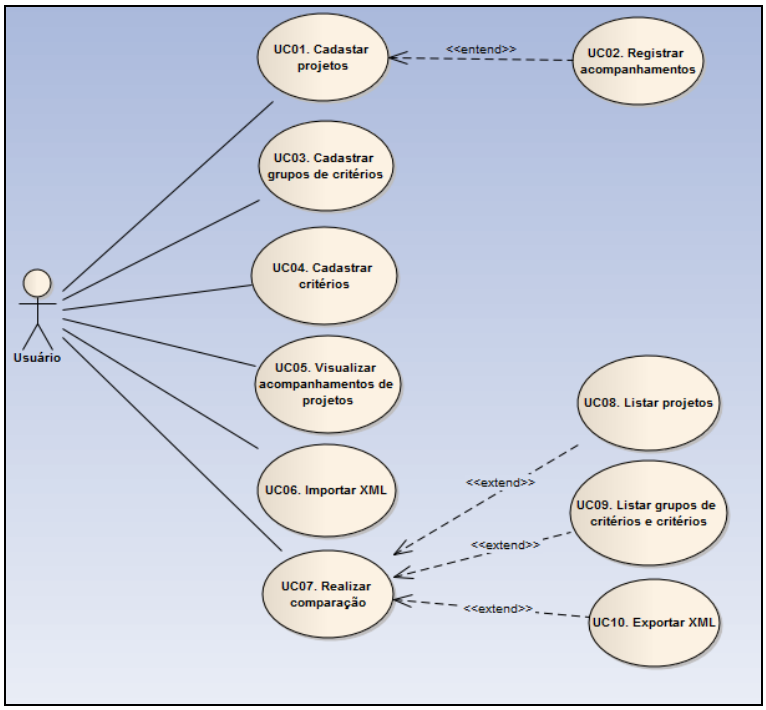


Figura 6 - Diagrama de casos de uso

A seguir são descritos os casos de uso. Nas descrições foram omitidas as funcionalidade de cancelar e sair da tela.

3.2.1.1 Cadastrar projetos

O primeiro caso de uso (Quadro 4), designado `Cadastrar projetos`, descreve como o usuário pode incluir os projetos da organização na aplicação. Além do cenário principal o caso de uso possui dois cenários alternativos que permitem alterar e excluir o projeto. O caso de uso possui um cenário de exceção caso o usuário não informe o nome de projeto.

UC01 – Cadastrar projetos: possibilita ao usuário uma forma de incluir, alterar e excluir os projetos que a organização possui.	
Requisitos atendidos	RF01.
Pré-condições	Não possui.
Cenário principal	1) Usuário acessa o sistema de seleção e priorização de projetos. 2) Usuário seleciona a opção para cadastrar projeto. 3) Usuário informa os dados do projeto. 4) Usuário seleciona a opção parar salvar o projeto. 5) Sistema salva o projeto.
Fluxo alternativo 01	No passo 3 do cenário principal, caso o usuário desejar alterar o projeto, deve-se selecionar um projeto já existente. 1) Usuário busca o projeto que deseja alterar. 2) Usuário altera as informações do projeto. 3) Usuário seleciona a opção parar salvar o projeto. 4) Sistema salva o projeto
Fluxo alternativo 02	No passo 3 do cenário principal, caso o usuário desejar excluir o projeto, deve-se selecionar um projeto já existente. 1) Usuário busca o projeto que deseja alterar. 2) Usuário seleciona a opção parar excluir o projeto. 3) Sistema exclui o projeto
Fluxo de exceção	Se no passo 3 do cenário principal não for informado o nome do projeto, será gerada uma mensagem de alerta pelo sistema.
Pós-condições	Não possui.

Quadro 4 - Caso de uso `UC01`

3.2.1.2 Registrar acompanhamentos

O segundo caso de uso (Quadro 5), designado `Registrar acompanhamentos`, descreve como o usuário pode incluir o acompanhamento de um projeto. Além do cenário principal o caso de uso possui dois cenários alternativos que permitem alterar e excluir os acompanhamentos. O caso de uso possui um cenário de exceção caso o usuário não informe a descrição do acompanhamento.

UC02 – Registrar acompanhamentos: possibilita ao usuário uma forma de incluir, alterar e excluir os acompanhamentos de projetos que a organização possui.	
Requisitos atendidos	RF10.
Pré-condições	Possuir o projeto cadastrado.
Cenário principal	1) Usuário acessa o sistema de seleção e priorização de projetos. 2) Usuário seleciona a opção para cadastrar o projeto. 3) Usuário seleciona o projeto que deseja realizar o acompanhamento. 4) Usuário seleciona o botão acompanhamento. 5) Usuário informa os dados do acompanhamento. 6) Usuário seleciona a opção parar salvar o acompanhamento. 7) Sistema salva o acompanhamento.
Fluxo alternativo 01	No passo 5 do cenário principal, caso o usuário desejar alterar o acompanhamento, deve-se selecionar um acompanhamento já existente. 1) Usuário seleciona um acompanhamento. 2) Usuário altera o acompanhamento. 3) Usuário seleciona a opção parar salvar o acompanhamento. 4) Sistema salva o acompanhamento.
Fluxo alternativo 02	No passo 5 do cenário principal, caso o usuário desejar excluir o acompanhamento, deve-se selecionar um acompanhamento já existente. 1) Usuário seleciona um acompanhamento. 1) Usuário seleciona a opção parar excluir o acompanhamento. 2) Sistema exclui o acompanhamento.
Fluxo de exceção	Se no passo 5 do cenário principal não for informado a descrição do acompanhamento, será gerada uma mensagem de alerta pelo sistema.
Pós-condições	Não possui.

Quadro 5 - Caso de uso `UC01`

3.2.1.3 Cadastrar grupos de critérios

O terceiro caso de uso (Quadro 6), designado `Cadastrar grupos de critérios`, descreve como o usuário pode incluir os grupos de critérios da organização na aplicação. Além do cenário principal o caso de uso possui dois cenários alternativos que permitem alterar e excluir os grupos de critérios. O caso de uso possui um cenário de exceção caso o usuário não informe o nome de grupo de critério.

UC03 – Cadastrar grupos de critérios: possibilita ao usuário uma forma de incluir, alterar e excluir os grupos de critérios que a organização possui.	
Requisitos atendidos	RF02.
Pré-condições	Não possui.
Cenário principal	1) Usuário acessa o sistema de seleção e priorização de projetos. 2) Usuário seleciona a opção para cadastrar grupos de critérios. 3) Usuário informar o nome do grupo de critério. 4) Usuário seleciona a opção parar salvar o grupo de critério. 5) Sistema salva o grupo de critério.
Fluxo alternativo 01	No passo 3 do cenário principal, caso o usuário desejar alterar o grupo de critério, deve-se selecionar um grupo já existente. 1) Usuário seleciona um grupo de critério. 2) Usuário altera o nome do grupo de critério. 3) Usuário seleciona a opção parar salvar o grupo de critério. 4) Sistema salva o grupo de critério.
Fluxo alternativo 02	No passo 3 do cenário principal, caso o usuário desejar excluir o grupo de critério, deve-se selecionar um grupo já existente. 1) Usuário seleciona um grupo de critério. 2) Usuário seleciona a opção parar excluir o grupo de critério. 3) Sistema exclui o grupo de critério
Fluxo de exceção	Se no passo 3 do cenário principal não for informado o nome do grupo de critério, será gerada uma mensagem de alerta pelo sistema.
Pós-condições	Não possui.

Quadro 6 - Caso de uso `UC03`

3.2.1.4 Cadastrar critérios

O quarto caso de uso (Quadro 7), designado `Cadastrar critérios`, descreve como o usuário pode incluir os critérios da organização na aplicação. Além do cenário principal o caso de uso possui dois cenários alternativos que permitem alterar e excluir os critérios. O caso de uso possui dois cenários de exceções para caso o usuário não informe o nome do critério ou o grupo de critério associado.

UC04 – Cadastrar critérios: possibilita ao usuário uma forma de incluir, alterar e excluir os critérios que a organização possui.	
Requisitos atendidos	RF02.
Pré-condições	Possuir o grupo de critério já cadastrado.
Cenário principal	1) Usuário acessa o sistema de seleção e priorização de projetos. 2) Usuário seleciona a opção para cadastrar critérios. 3) Usuário informa o nome do critério e o grupo de critério a qual o critério está associado. 4) Usuário seleciona a opção parar salvar o critério. 5) Sistema salva o critério.
Fluxo alternativo 01	No passo 3 do cenário principal, caso o usuário desejar alterar o critério, deve-se selecionar um critério já existente 1) Usuário seleciona o critério. 2) Usuário altera o critério. 3) Usuário seleciona a opção parar salvar o critério. 4) Sistema salva o critério.
Fluxo alternativo 02	No passo 3 do cenário principal, caso o usuário desejar excluir o critério, deve-se selecionar um critério já existente. 1) Usuário seleciona o critério. 2) Usuário seleciona a opção parar excluir o critério. 3) Sistema exclui o critério.
Fluxo de exceção 01	Se no passo 3 do cenário principal caso não for informado o nome do critério, será gerada uma mensagem de alerta pelo sistema.
Fluxo de exceção 02	Se no passo 3 do cenário principal não for informado o grupo de critério, será gerada uma mensagem de alerta pelo sistema.
Pós-condições	Não possui.

Quadro 7 - Caso de uso UC04

3.2.1.5 Visualizar acompanhamentos de projetos

O quinto caso de uso (Quadro 8), designado `Visualizar acompanhamentos de projetos`, descreve como o usuário pode gerar um relatório para visualizar os acompanhamentos realizados nos projetos. O caso de uso não possui fluxos de exceção e possui um fluxo alternativo.

UC05 – Visualizar acompanhamentos de projetos: possibilita ao usuário gerar um relatório com o acompanhamento de todos os projetos.	
Requisitos atendidos	RF13.
Pré-condições	Possuir o acompanhamento cadastrado.
Cenário principal	1) Usuário acessa o sistema de seleção e priorização de projetos. 2) Usuário seleciona a opção para gerar o relatório com o histórico dos acompanhamentos dos projetos. 3) Sistema abre uma tela com a listagem dos históricos de acompanhamentos dos projetos. 4) Usuário seleciona a opção Salvar. 5) Sistema salva a listagem em um arquivo PDF.
Fluxo alternativo 01	No passo 4 do cenário principal, caso o usuário não desejar salvar o histórico de acompanhamentos. 1) Usuário seleciona a opção Sair. 2) Sistema fecha a listagem de acompanhamentos.
Pós-condições	Não possui.

Quadro 8 - Caso de uso `UC05`

3.2.1.6 Importar XML

O sexto caso de uso (Quadro 9), designado `Importar XML`, descreve como o usuário pode importar um arquivo XML para realizar a comparação entre os projetos. O caso de uso não possui fluxos de exceção e fluxo alternativo.

UC06 – Importar XML: possibilita ao usuário uma forma de importar um arquivo XML.	
Requisitos atendidos	RF12.
Pré-condições	Possuir o arquivo XML para realizar a importação.
Cenário principal	1) Usuário acessa o sistema de seleção e priorização de projetos. 2) Usuário seleciona a opção para importar o arquivo XML. 3) Usuário seleciona o arquivo XML que deseja importar. 4) Sistema importar o arquivo XML e abre a tela de realizar comparação entre projetos.
Pós-condições	Não possui.

Quadro 9 - Caso de uso `UC06`

3.2.1.7 Realizar comparação

O sétimo caso de uso (Quadro 10), designado `Realizar comparação`, descreve como o sistema irá realizar a comparação entre os projetos selecionados pelo usuário. O caso de uso possui cinco fluxos de exceções e não possui fluxo alternativo.

UC07 – Realizar comparação entre projetos: possibilita a comparação entre os projetos selecionados.	
Requisitos atendidos	RF03, RF04, RF05, RF06 e RF07.
Pré-condições	Possuir cadastrado no mínimo dois projetos. Possuir cadastrado no mínimo dois grupos de critérios. Possuir cadastrado no mínimo dois critérios.
Cenário principal	1) Usuário acessa o sistema de seleção e priorização de projetos. 2) Usuário seleciona a opção para realizar a comparação entre projetos. 3) Sistema apresenta a tela para realizar a comparação entre os projetos. 4) Usuário seleciona os projetos que deseja priorizar. 5) Usuário seleciona os grupos de critério que serão analisados. 6) Usuário seleciona os critério que serão analisados. 7) Usuário informa o peso entre os grupos de critérios selecionados. 8) Usuário informa o peso entre os critérios selecionados. 9) Sistema calcula a priorização entre os grupos de critérios e critérios selecionados. 10) Usuário informa o peso entre os projetos selecionados para cada grupo de critério escolhido. 11) Usuário informa o peso entre os projetos selecionados para cada critério escolhido. 12) Sistema apresenta a priorização dos projetos selecionados.
Fluxo de exceção 1	No passo 4, caso o usuário selecionar menos que dois ou mais que dez projetos, será gerar uma mensagem de alerta pelo sistema.
Fluxo de exceção 2	No passo 5, caso o usuário selecionar menos que dois ou mais que dez grupos de critérios, será gerar uma mensagem de alerta pelo sistema.
Fluxo de exceção 3	No passo 6, caso o usuário selecionar menos que dois ou mais que dez critérios, será gerar uma mensagem de alerta pelo sistema.
Fluxo de exceção 4	No passo 9, caso o valor da taxa de consistência não for adequado, será gerada uma mensagem de alerta pelo sistema.
Fluxo de exceção 5	No passo 12 caso o valor da taxa de consistência não for adequado, será gerada uma mensagem de alerta pelo sistema.
Pós-condições	O sistema apresenta as opções para gerar o relatório com a lista dos projetos, dos grupos de critérios e critérios e exportar para arquivo XML.

Quadro 10 - Caso de uso `UC07`

3.2.1.8 Listar projetos

O oitavo caso de uso (Quadro 11), designado `Listar projetos`, descreve como o usuário pode gerar um relatório com a lista dos projetos ordenados por sua priorização. O caso de uso não possui um fluxo de exceção e possui um fluxo alternativo.

UC08 – Listar projetos: possibilita ao usuário gerar um relatório com a priorização dos projetos selecionados.	
Requisitos atendidos	RF08.
Pré-condições	Ter realizada a comparação entre pelo menos dois projetos.
Cenário principal	1) Usuário seleciona a opção para gerar o relatório com a lista dos projetos selecionados. 2) Sistema abre uma tela com a listagem dos projetos com a sua priorização. 3) Usuário seleciona a opção Salvar. 4) Sistema salva a listagem em um arquivo PDF.
Fluxo alternativo 01	No passo 2 do cenário principal, caso o usuário não desejar salvar a lista de projetos. 1) Usuário seleciona a opção Sair. 2) Sistema fecha a listagem de projetos.
Pós-condições	Não possui.

Quadro 11 - Caso de uso `UC08`

3.2.1.9 Listar grupos de critérios e critérios

O nono caso de uso (Quadro 12), designado `Listar grupos de critérios e critérios`, descreve como o usuário pode gerar um relatório com a lista dos critérios ordenados por sua priorização. O caso de uso não possui um fluxo de exceção e possui um fluxo alternativo.

UC09 – Listar grupos de critérios e critérios: possibilita ao usuário gerar um relatório com a priorização dos grupos de critérios e critérios selecionados.	
Requisitos atendidos	RF09.
Pré-condições	Ter realizada a comparação entre pelo menos dois projetos.
Cenário principal	1) Usuário seleciona a opção para gerar o relatório com a lista dos grupos de critérios e critérios selecionados. 2) Sistema abre uma tela com a listagem dos grupos de critérios e critérios com os seus pesos. 3) Usuário seleciona a opção Salvar. 4) Sistema salva a listagem em um arquivo PDF.
Fluxo alternativo 01	No passo 2 do cenário principal, caso o usuário não desejar salvar a lista de grupos de critérios e critérios. 1) Usuário seleciona a opção Sair. 2) Sistema fecha a listagem de grupos de critérios e critérios.
Pós-condições	Não possui.

Quadro 12 - Caso de uso `UC09`

3.2.1.10 Exportar XML

O décimo caso de uso (Quadro 13), designado `Exportar XML`, descreve como o

usuário pode exportar a comparação entre os projetos realizada para um arquivo XML. O caso de uso não possui um fluxo de exceção e fluxo alternativo.

UC10 – Exportar XML: possibilita ao usuário exportar as informações da comparação para um arquivo XML.	
Requisitos atendidos	RF11.
Pré-condições	Ter realizada a comparação entre pelo menos dois projetos.
Cenário principal	1) Usuário seleciona a opção para exportar para XML. 2) Sistema abre uma tela para escolher o diretório que deseja salvar o arquivo e informar o nome que será utilizado. 3) Usuário seleciona a diretório, informa o nome e seleciona a opção Salvar. 4) Sistema salva o arquivo XML no diretório escolhido.
Pós-condições	Não possui.

Quadro 13 - Caso de uso `UC10`

3.2.2 Diagrama de atividades

A aplicação de apoio à priorização e seleção de projetos possui dez atividades (Figura 7), sendo que sete destas atividades são executadas pelo ator `Usuário`, que representa a pessoa que faz uso da aplicação e três são executadas pelo sistema.

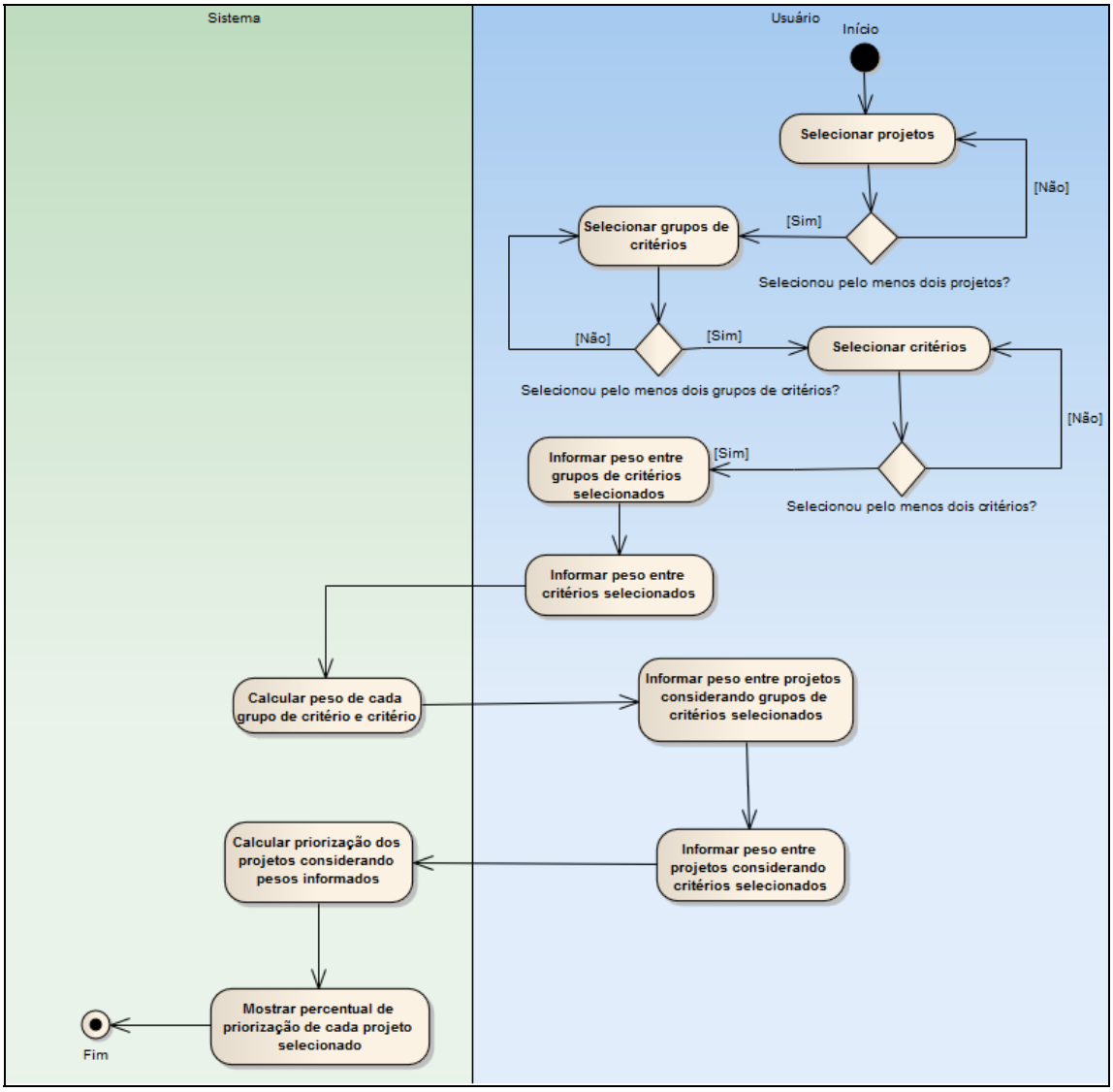


Figura 7 - Diagrama de atividades

Primeiramente o ator `Usuário` do sistema escolhe os projetos que deseja selecionar, os grupos de critérios e critérios que serão avaliados. Em seguida o `Usuário` informa os pesos entre os grupos selecionados e os critérios selecionados. Com base nestes pesos, o `Sistema` calcula o peso que cada grupo de critério e cada critério irá possuir. O `Usuário` informa o peso entre os projetos considerando cada grupo de critério e critério selecionados. Por fim o `Sistema` calcula a priorização dos projetos e mostra o resultado final para o `Usuário`.

3.2.3 Diagrama de classes

O diagrama de classes apresenta uma visão de como as classes estão estruturadas e relacionadas. Nesta subseção são descritas as classes necessárias para o desenvolvimento da aplicação de seleção e priorização de projetos. Nesse contexto, para uma visualização geral, é apresentado na Figura 8 o diagrama conceitual da aplicação. Neste diagrama são apresentados os relacionamentos que os principais recursos da aplicação possuem e o local onde as informações são salvas, se no banco de dados, denominado entidade, ou se no arquivo XML.

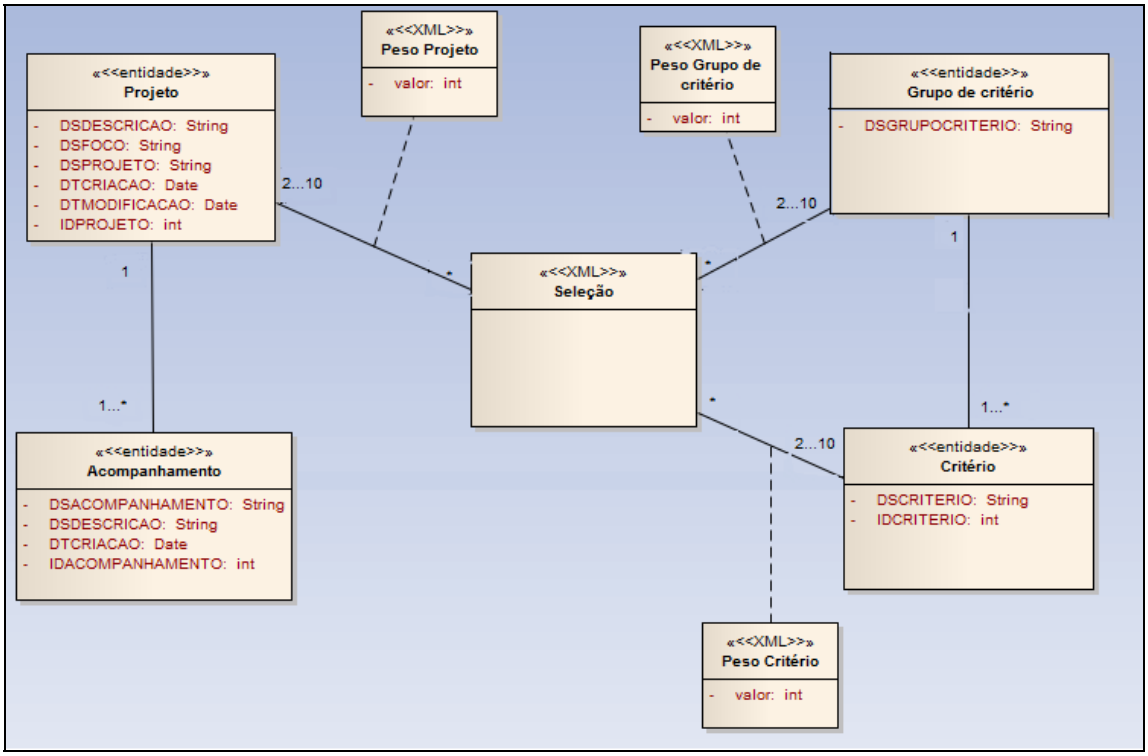


Figura 8 - Diagrama conceitual

Para favorecer o melhor entendimento, as classes estão reunidas, conforme as ligações lógicas entre si, em pacotes (Figura 9). Nessa subseção serão abordadas apenas as principais funcionalidades de cada classe. Os detalhes de cada classe serão abordados na subseção 3.3.2.

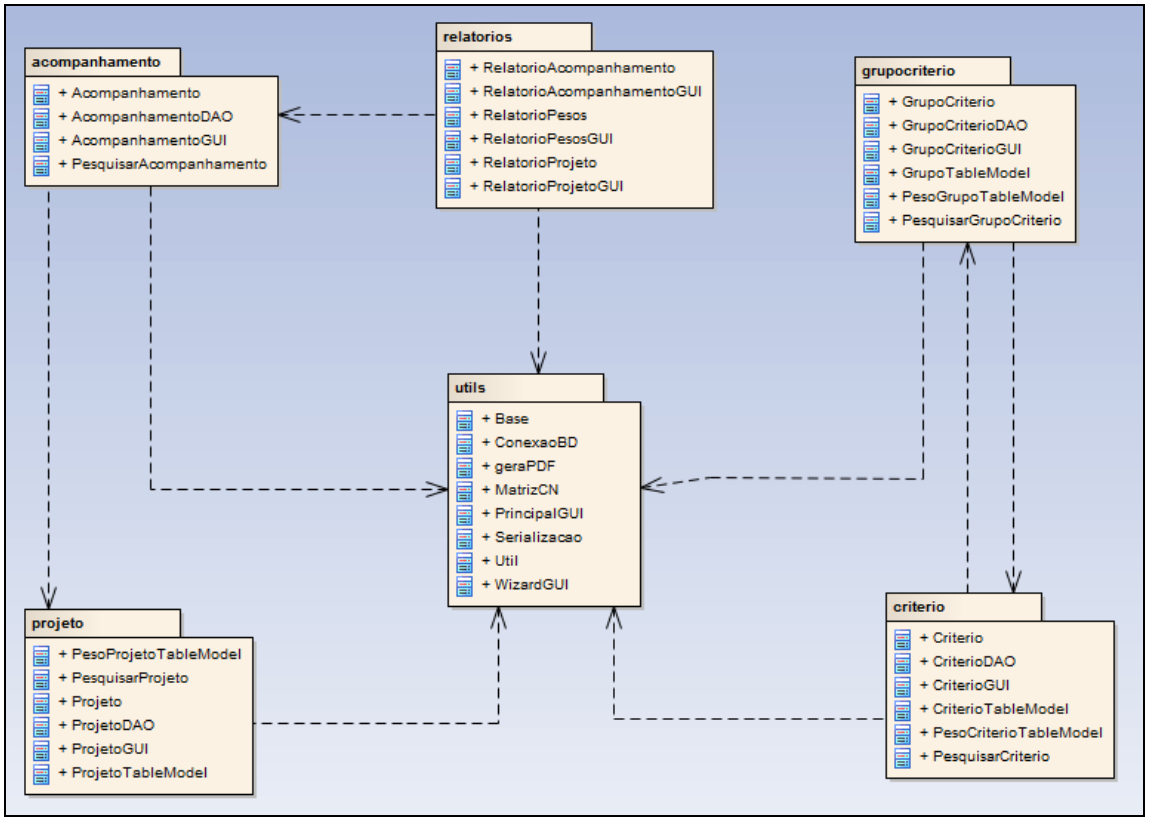


Figura 9 - Pacotes da aplicação

3.2.3.1 Pacote `utils`

O primeiro pacote é denominado `utils` (Figura 10) e possui todas as classes que são utilizadas pelas demais classes da aplicação. Fazem parte deste pacote as classes `Base`, `ConexaoBD`, `geraPDF`, `MatrizCN`, `PrincipalGUI`, `Serializacao`, `Util` e `WizardGUI`.

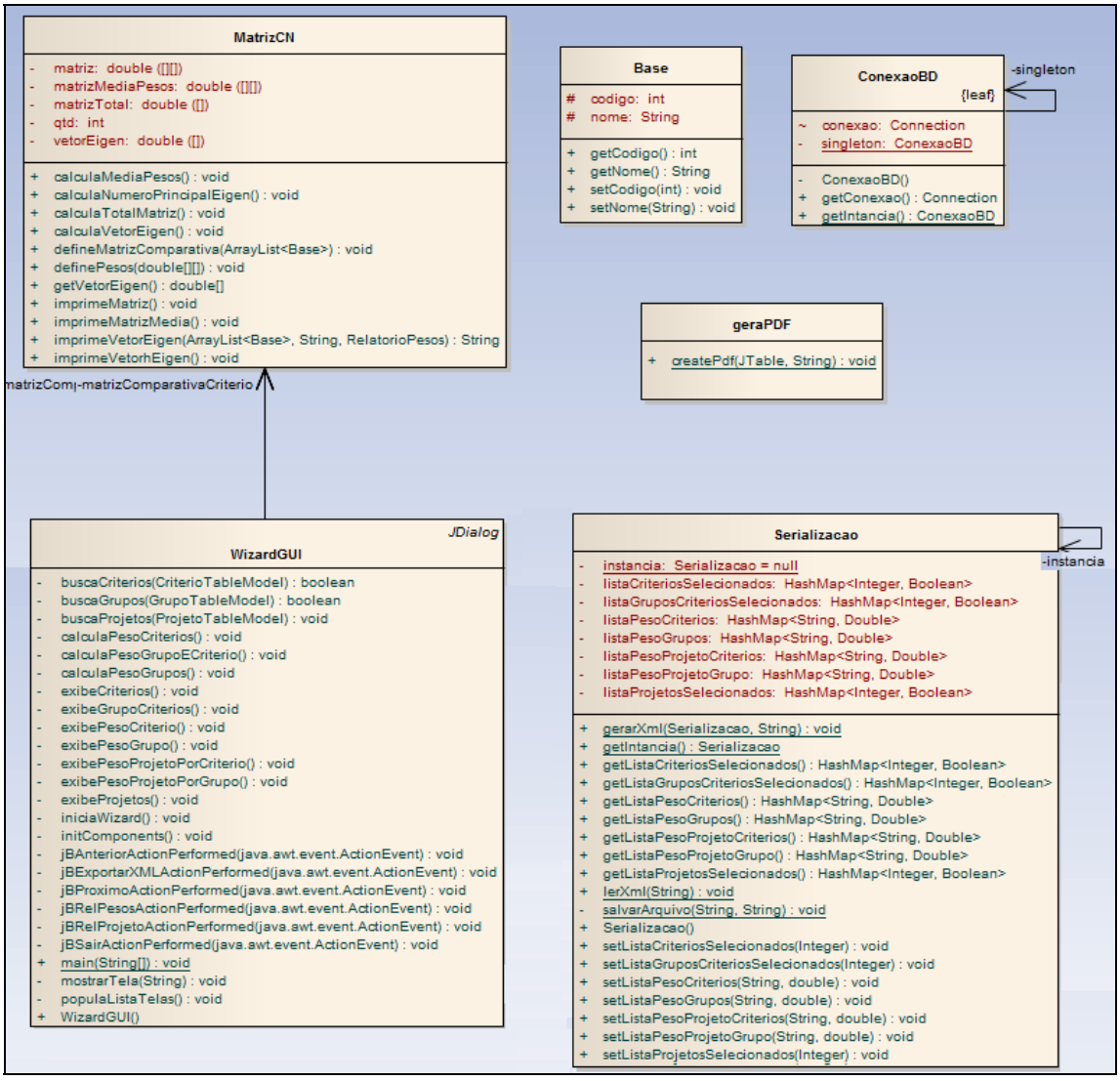


Figura 10 - Pacotes `utils`

A classe `Base` possui os atributos código e nome. Essa classe foi criada, pois os métodos que realizam os cálculos na comparação dos projetos são globais e por isso as classes que realizam os cálculos podem receber uma lista com os grupos de critérios ou critérios selecionados. Assim, a classe sempre irá receber uma lista do tipo `Base`, que pode conter um grupo de critério ou um critério.

A classe `ConexaoBD` foi criada apenas para realizar a conexão com o banco de dados. Essa classe é utilizada por todas as classes que precisar alterar/excluir ou buscar alguma informação do banco de dados.

A classe `geraPDF` é responsável por gerar os arquivos PDF de acordo com o relatório escolhido no sistema.

A classe `MatrizCN` é responsável por realizar todos os cálculos durante a comparação dos projetos.

A classe `PrincipalGUI` contêm o menu do sistema com as suas cinco funcionalidades.

A classe `Serializacao` é responsável por guardar as informações que serão exportadas e importadas de um arquivo XML (*eXtensible Markup Language*). Alguns métodos dessa classe estão sublinhados, conforme diagrama apresentado na Figura 10. Isso ocorre quando o método está declarado como sendo do tipo *static*.

A classe `Util` é responsável apenas pelo arredondamento dos valores dos cálculos efetuados.

Por fim a classe `WizardGUI` é responsável por toda a rotina de comparação dos projetos. Não foram apresentados no diagrama de classes os atributos desta classe, pois ela possui muitos atributos para renderização das telas que serão apresentadas durante a comparação dos projetos.

As explicações das principais classes deste pacote estão descritas na seção 3.3.2.4.

3.2.3.2 Pacote `acompanhamento`

O segundo pacote é denominado `acompanhamento` (Figura 10) e possui todas as classes que são utilizadas pela rotina de acompanhamento de projetos. Fazem parte deste pacote as classes `Acompanhamento`, `AcompanhamentoDAO`, `PesquisaAcompanhamneto`, e `AcompanhamentoGUI`.

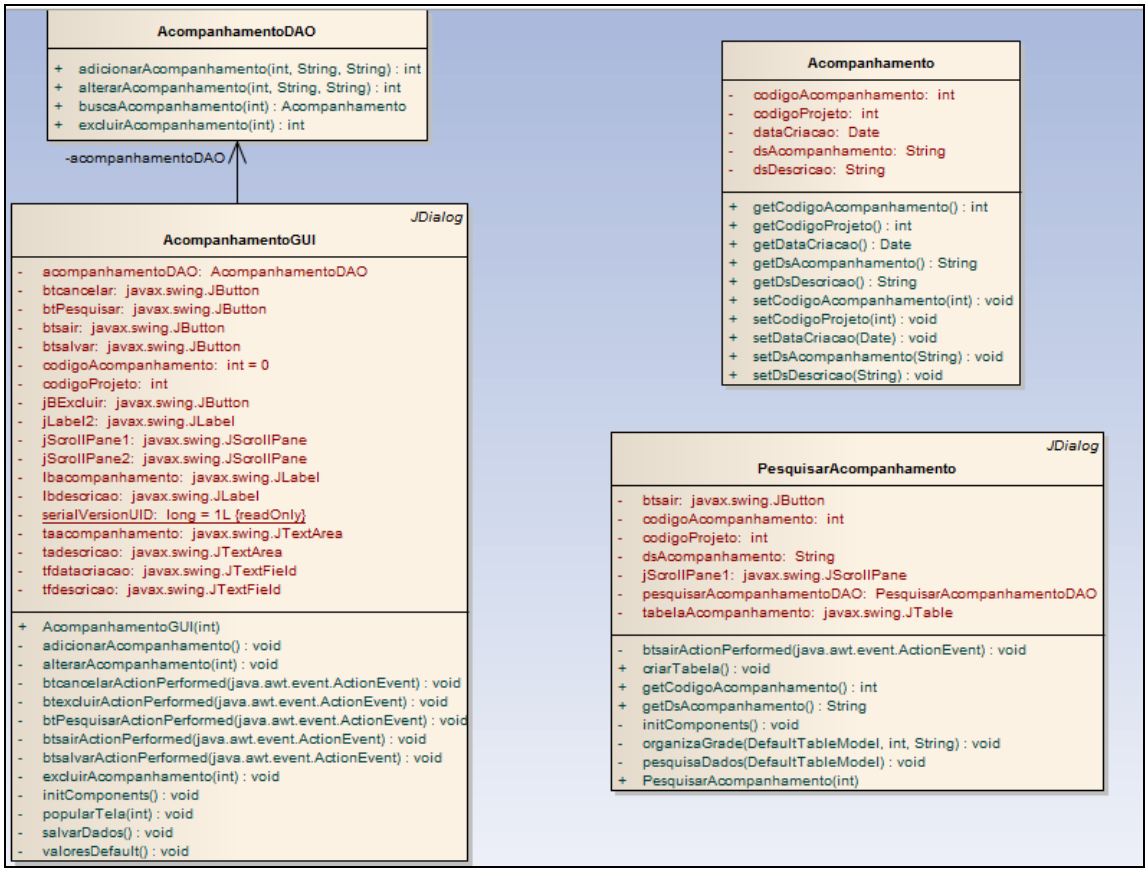


Figura 11 - Pacote `acompanhamento`

As explicações de todas as classes deste pacote estão descritas na seção 3.3.2.1.

3.2.3.3 Pacote `relatorio`

O pacote `relatório` (Figura 12) é responsável por todas as classes que são utilizadas na geração de relatórios do sistema. Fazem parte deste pacote as classes `RelatorioAcompanhamento`, `RelatorioAcompanhamentoGUI`, `RelatorioPesos`, `RelatorioPesosGUI`, `RelatorioProjeto` e `RelatorioProjetoGUI`.

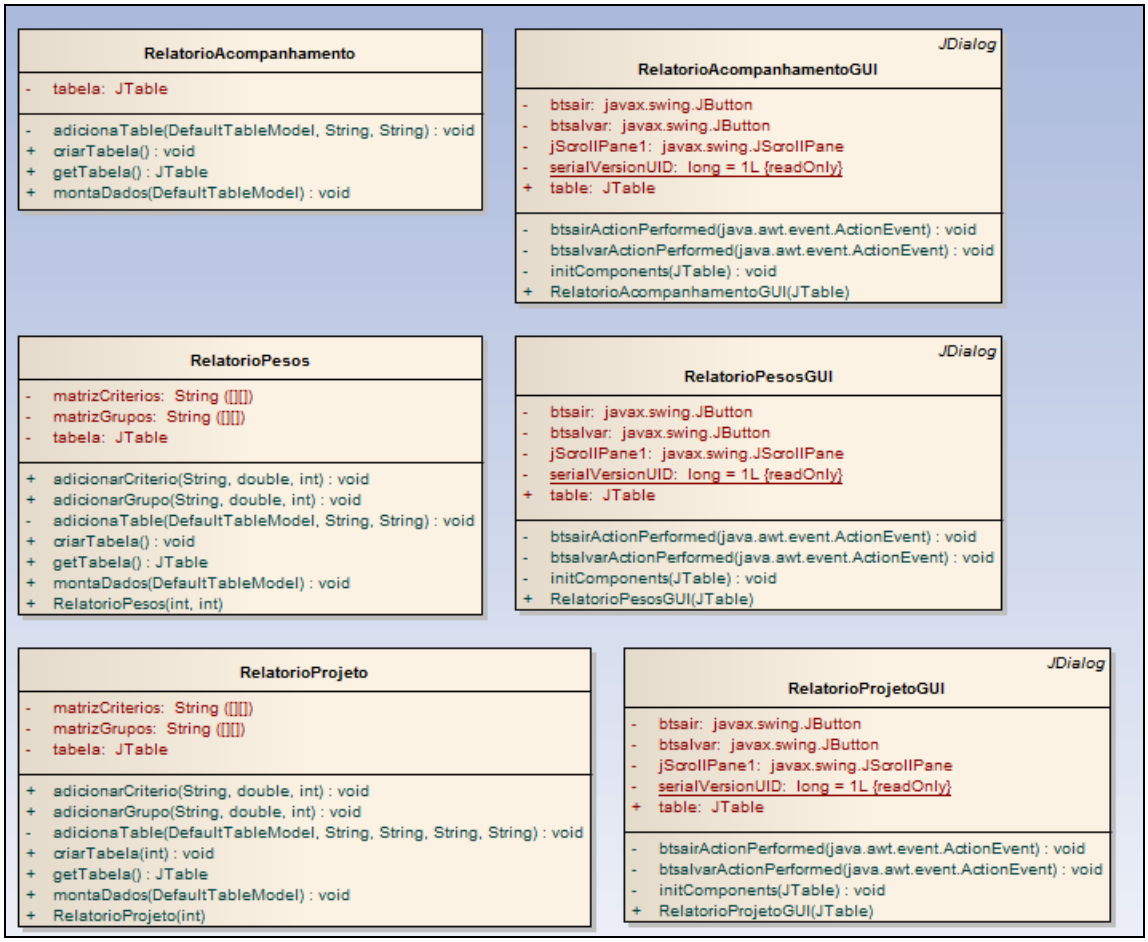


Figura 12 - Pacote **`relatório`**

As explicações de todas as classes deste pacote estão descritas nas seções 3.3.2.4.1, 3.3.2.4.2 e 3.3.2.5.

3.2.3.4 Pacote `grupocriterio`

O pacote `grupocriterio` (Figura 13) é responsável pelas funções de criar, alterar, excluir e pesquisar grupos de critérios. Fazem parte deste pacote as classes `GrupoCriterio`, `GrupoCriterioDAO`, `GrupoCriterioGUI`, `GrupoTableModel`, `PesoGrupoTableModel`, `PesquisarGrupoCriterio` e `PesquisarGrupoCriterioDAO`.

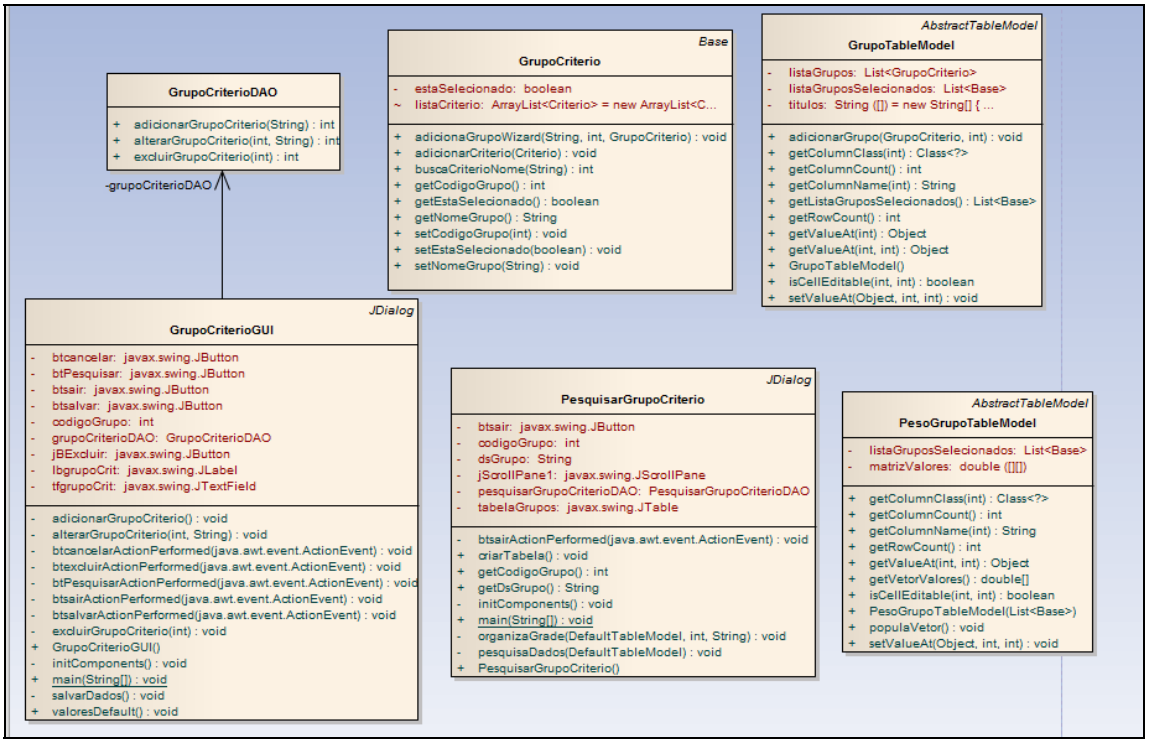


Figura 13 - Pacote `grupocriterio`

As explicações de todas as classes deste pacote estão descritas nas seções 3.3.2.2 e 3.3.2.4.

3.2.3.5 Pacote `projeto`

O pacote `projeto` (Figura 14) é responsável pelas funções de criar, alterar, excluir e pesquisar projetos. Fazem parte deste pacote as classes `Projeto`, `ProjetoDAO`, `ProjetoGUI`, `ProjetoTableModel`, `PesoProjetoTableModel`, `PesquisarProjeto` e `PesquisarProjetoDAO`.

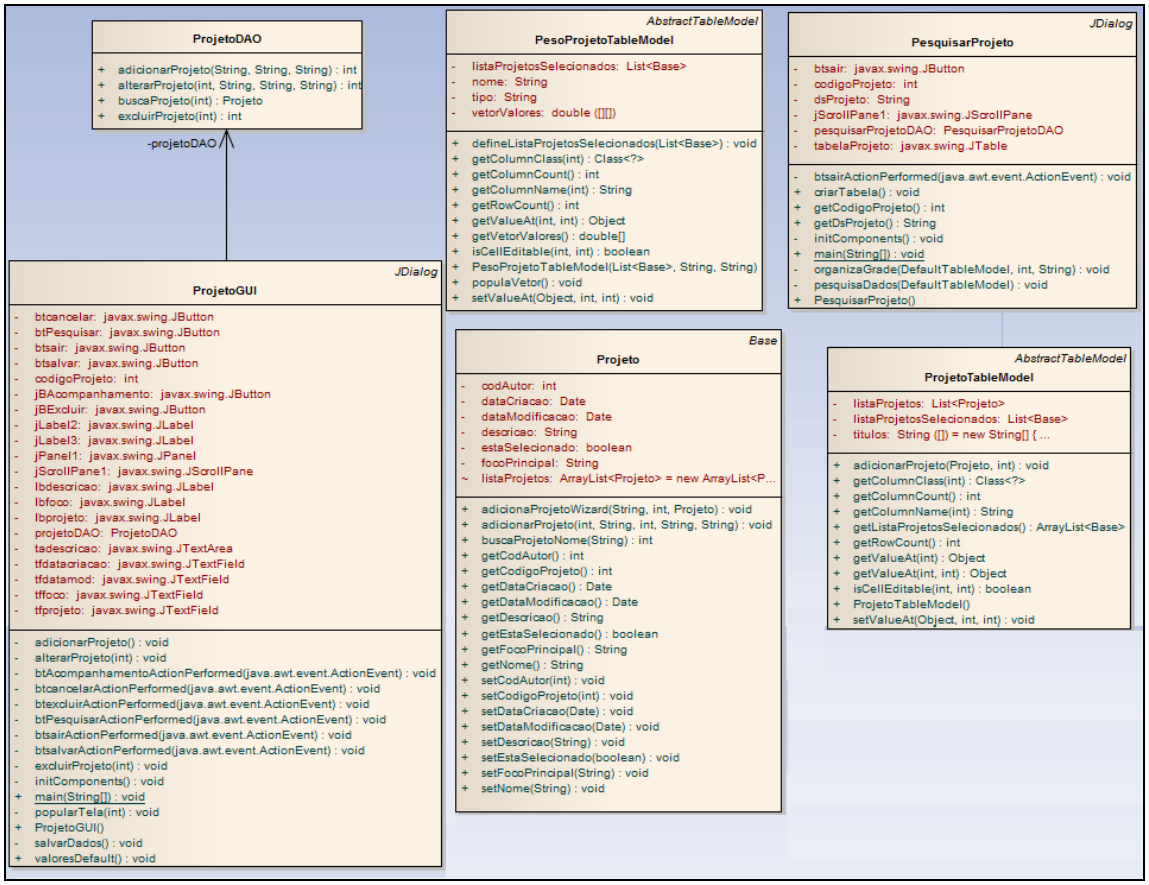


Figura 14 - Pacote `projeto`

As explicações de todas as classes deste pacote estão descritas nas seções 3.3.2.1 e 3.3.2.4.

3.2.3.6 Pacote `criterio`

O pacote `criterio` (Figura 15) é responsável pelas funções de criar, alterar, excluir e pesquisar critérios. Fazem parte deste pacote as classes `Criterio`, `CriterioDAO`, `CriterioGUI`, `CriterioTableModel`, `PesoCriterioTableModel`, `PesquisarCriterio` e `PesquisarCriterioDAO`.

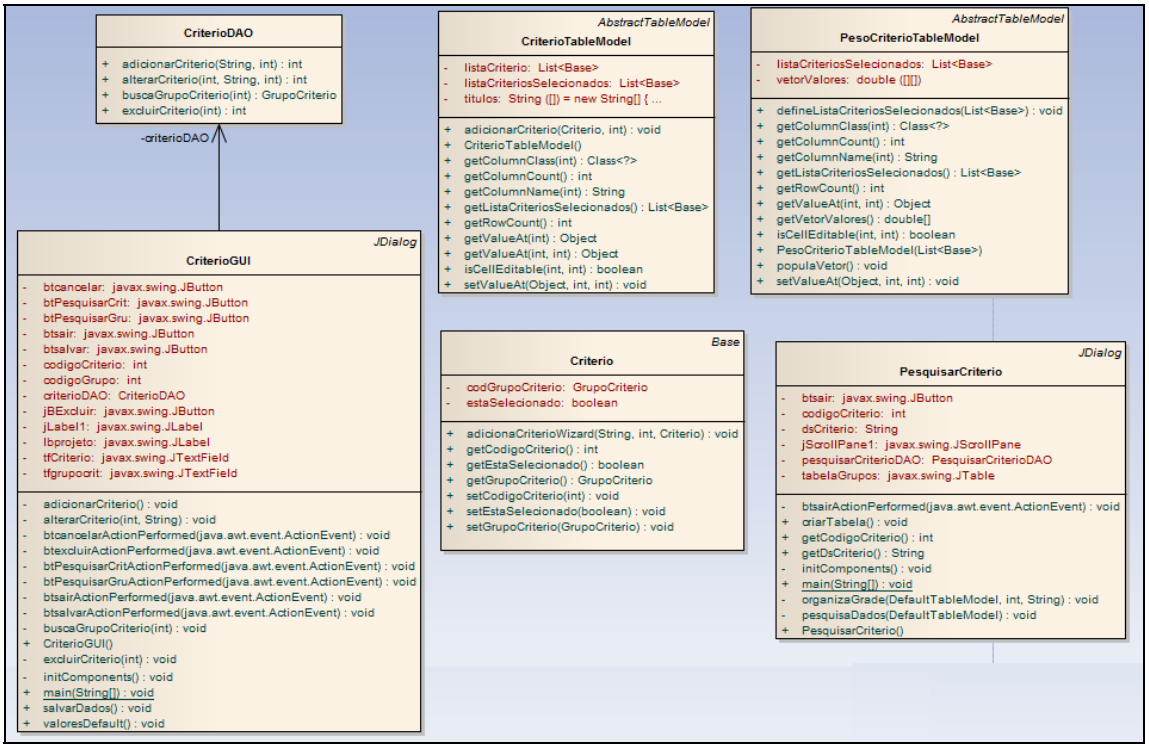


Figura 15 - Pacote `criterio`

As explicações de todas as classes deste pacote estão descritas nas seções 3.3.2.3 e 3.3.2.4.

3.2.4 Diagrama de seqüência

O diagrama de seqüência (Figura 16) apresenta uma visão interna do processo e da comunicação entre as classes para a rotina de `Realizar comparação entre projetos` (caso de uso `UC07`).

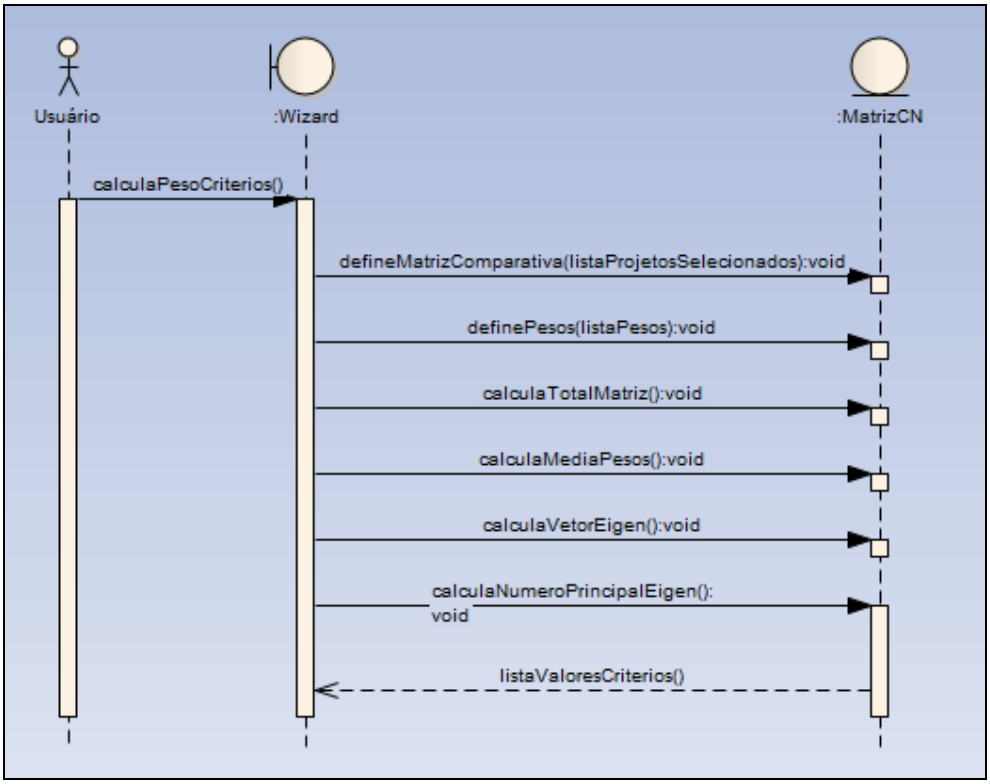


Figura 16 - Diagrama de seqüência

A interface `Wizard` recebe a solicitação do ator `Usuário` para realizar o cálculo do peso dos critérios. Por sua vez, a interface chama os métodos da classe `MatrizCN` para realizar os cálculos. Após a classe `MatrizCN` realizar os cálculos, ela retorna a lista com os valores dos critérios para a interface que apresenta os valores para o `Usuário`.

3.3 IMPLEMENTAÇÃO

A seguir são descritas as técnicas e ferramentas utilizadas na implementação, bem como detalhes das principais classes e rotinas implementadas durante o desenvolvimento da aplicação de seleção e priorização de projetos.

3.3.1 Técnicas e ferramentas utilizadas

A aplicação de seleção e priorização de projetos foi implementada sob o paradigma da

orientação a objetos na linguagem programação Java, utilizando-se o ambiente de desenvolvimento Eclipse na versão *Indigo Service Release* 1.

Já o desenvolvimento do leiaute utilizado para a modelagem foi realizado com o auxílio do NetBeans IDE na versão 6.8.

3.3.2 Operacionalidade da implementação

Nesta subseção é apresentado um estudo de caso das funcionalidades da aplicação. Este estudo de caso é feito através da seleção e priorização de projeto da área de TI.

A aplicação tem como objetivo selecionar e priorizar os projetos dentro da organização aplicando a técnica *AHP*. Na aplicação serão cadastrados os projetos a serem priorizados, os grupos de critérios e critérios que serão utilizados para realizar a priorização, bem como os pesos entre os projetos, grupos de critérios e critérios que serão utilizados para chegar ao resultado final.

O primeiro passo para utilizar a aplicação é executar a classe `PrincipalGUI` que é responsável pela visualização do menu da aplicação. O menu possui atalhos para as funcionalidades da aplicação (Figura 17).

Figura 17 - Menu da aplicação

3.3.2.1 Cadastrar projeto

A primeira funcionalidade é cadastrar projeto (caso de uso `UC01`) e é responsável pela

criação, alteração e exclusão dos projetos. A funcionalidade é composta por três classes principais `ProjetoGUI`, `ProjetoDAO`, `PesquisarProjeto` e `Projeto`. A classe `ProjetoGUI` é responsável pelo leiaute da tela de cadastro (Figura 18) e faz as chamadas das classes `ProjetoDAO` e `Projeto`.

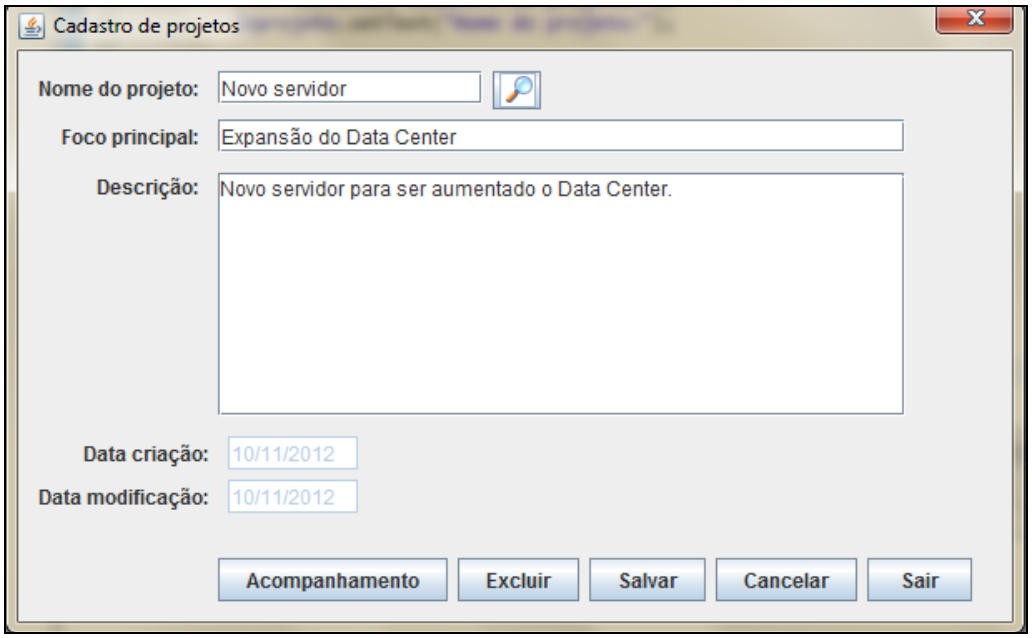


Figura 18 - Cadastro de projeto

Por exemplo, ao informar os dados de um novo projeto e selecionar o botão `Salvar`, a classe `ProjetoGUI` chama o método `adicionarProjeto`. Esse método executa o método `adicionarProjeto` da classe `ProjetoDAO`, passando por parâmetro o nome do projeto, o foco e a descrição do projeto informados na tela (Quadro 14).

```
private void adicionarProjeto() {

int retorno = projetoDAO.adicionarProjeto(tfprojeto.getText(),
tffoco.getText(), tadescricao.getText());

   if(retorno == -1){
       JOptionPane.showMessageDialog(null,
                                  "Erro ao tentar adicionar o projeto!");
   }else{
       JOptionPane.showMessageDialog(null,
                                  "Gravação na tabela projeto realizado com
sucesso!");
   }

}
```

Quadro 14 - Adicionar um novo projeto – classe ProjetoGUI

O método `adicionarProjeto` da classe `ProjetoDAO` por sua vez cria uma conexão com o banco de dados e executa o comando de inserção no banco de dados. O código do projeto é gerado pelo banco com base na *sequence* `S_ID_PROJETO`. Já a data da criação e data de modificação do projeto são geradas com base na função `currentTimeMillis` do Java (Quadro

15).

```
public int adicionarProjeto(String parmDsProjeto, String parmFoco, String
parmDescricao) {
	Connection connection;
	connection = ConexaoBD.getIntancia().getConexao();

	String sqlinsert = "insert into Projeto (IDPROJETO, DSPROJETO, DSFOCO,
DSDESCRICAO, DTCRIACAO, DTMODIFICACAO)"
			+ "values(S_ID_PROJETO.nextval,?,?,?,?,?)";

	try {
		PreparedStatement pstmt = connection.prepareStatement(sqlinsert);
		pstmt.setString(1, parmDsProjeto);
		pstmt.setString(2, parmFoco);
		pstmt.setString(3, parmDescricao);
		pstmt.setString(4, new SimpleDateFormat("dd/MM/yyyy")
				.format(new Date(System.currentTimeMillis())));
		pstmt.setString(5, new SimpleDateFormat("dd/MM/yyyy")
				.format(new Date(System.currentTimeMillis())));

		pstmt.execute();
		pstmt.close();

		return 0;
	} catch (SQLException e) {
		return -1;
	}

}
```

Quadro 15 - Adicionar um novo projeto – classe `ProjetoDAO`

Para realizar a conexão com o banco de dados, foi criada a classe `ConexaoBD` que utiliza o padrão de projeto `Singleton`. Foi aplicado esse padrão de projeto para garantir que exista apenas uma instância de conexão ao banco de dados (Quadro 16).

```
private ConexaoBD(){
	conexao = null;

	try {
		Class.forName("oracle.jdbc.driver.OracleDriver");
		conexao = DriverManager.getConnection(
				"jdbc:oracle:thin:@localhost:1521:XE",
"admin", "admin");
	} catch (ClassNotFoundException e) {
			e.printStackTrace();
	} catch (SQLException e) {
			e.printStackTrace();
	}
}

public static ConexaoBD getInstancia() {
	if (singleton == null) {
		singleton = new ConexaoBD();
	}
	return singleton;
}

public Connection getConexao(){
	return this.conexao;
}
```

Quadro 16 - Classe que faz a conexão com o banco de dados

Todas as classes que fazem manipulações dos dados no banco de dados possuem no seu nome o sufixo *Data Access Object* (DAO). O DAO é um padrão de projeto que abstrai e encapsula os mecanismos de acesso a dados escondendo os detalhes da execução da origem dos dados e é responsável por buscar os dados do banco de dados e converter em objetos para ser usado pela aplicação. Semelhantemente, deve saber como buscar os objetivos da aplicação, converter em instruções SQL e enviar para o banco de dados. Esse padrão de projeto foi utilizado em todas as rotinas que possuem alguma interação com o banco de dados (SARDAGNA; VAHLDICK, 2008).

A classe `Projeto` possui todos os atributos do projeto, juntamente com os métodos para atribuir valor e retornar o valor dos atributos. Essa classe é utilizada na tela de pesquisa dos projetos e pela rotina de comparação de projetos que serão apresentadas posteriormente.

Na tela de cadastro de projetos (Figura 18), ao lado do campo `Nome do projeto`, existe um botão para realizar a pesquisa dos projetos. Ao selecionar esse botão, a classe `ProjetoGUI` cria um novo objeto da classe `PesquisarProjeto`. Quando esse objeto é criado, é aberta uma nova tela com o código e nome de todos os projetos que foram cadastrados (Figura 19).

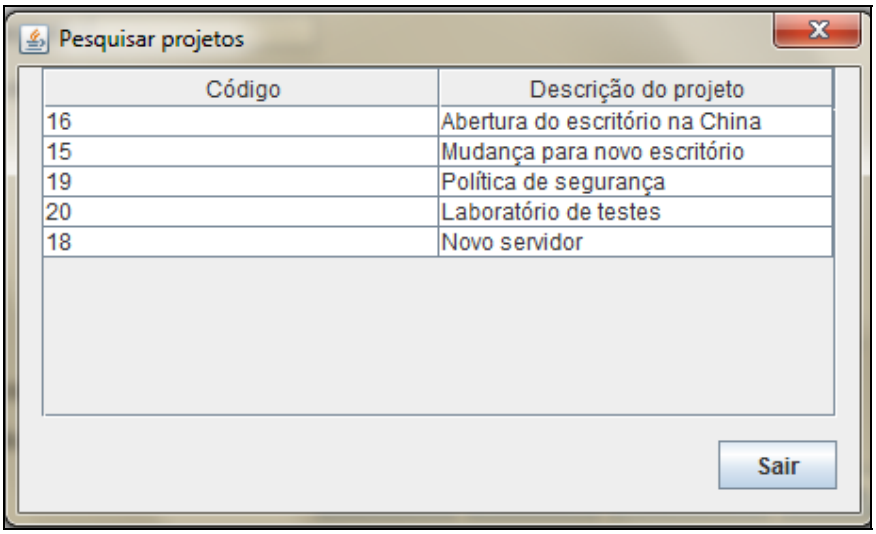


Figura 19 - Tela de pesquisa de projetos

A busca desses dados é feita através da classe `PesquisarProjetoDAO` que faz essa busca no banco de dados (Quadro 17).

```
public ArrayList<Projeto> buscaListaGrupoCriterios() throws Exception {

	try {
		Connection connection;
		ConexaoBD conexao = ConexaoBD.getInstancia();
		connection = conexao.getConexao();
		Statement stm = connection.createStatement();
		resultPesquisa = stm.executeQuery("select * from projeto");

		while (resultPesquisa.next()) {
			Projeto novoProjeto = new Projeto();

	novoProjeto.setCodigoProjeto(Integer.parseInt(resultPesquisa.getString("ID
PROJETO")));

	novoProjeto.setNome(resultPesquisa.getString("DSPROJETO"));
				listaProjetos.add(novoProjeto);
		}

	} catch (SQLException ex) {
		return listaProjetos;
	}
	return listaProjetos;
}
```

Quadro 17 - Buscar lista dos projetos

No cadastro de projetos também possui um recurso para realizar o acompanhamento dos projetos. Esse recurso possibilita o monitoramento das atividades que estão sendo executadas no projeto. O cadastro do acompanhamento possui a informação com a data de criação, que utiliza a data da máquina no momento do cadastro e uma descrição e o acompanhamento informado pelo `Usuário`, conforme pode ser visto na Figura 20.

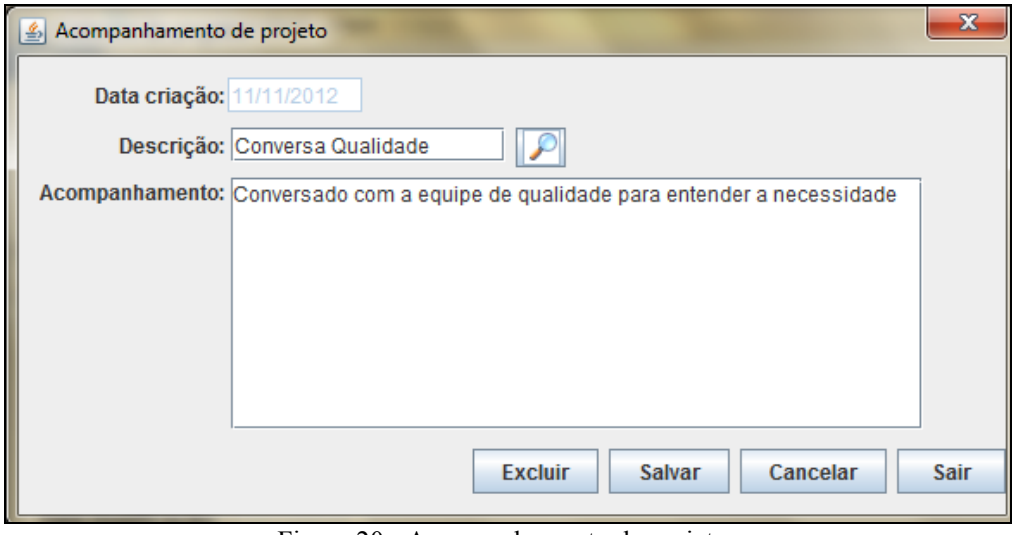


Figura 20 - Acompanhamento de projetos

O cadastro dos acompanhamentos possui os mesmos recursos que o cadastro dos projetos e funciona da mesma maneira explicada na seção 3.3.2.1.

3.3.2.2 Cadastrar grupo de critério

A segunda funcionalidade disponível na aplicação é cadastrar grupo de critérios (caso de uso `UC02`) e é responsável pela criação, alteração e exclusão dos grupos de critérios. A funcionalidade é composta por três classes principais `GrupoCriterioGUI`, `GrupoCriterioDAO` e `GrupoCriterio`. A classe `GrupoCriterioGUI` é responsável pelo leiaute da tela de cadastro (Figura 21) e faz as chamadas das classes `GrupoCriterioDAO`, `PesquisarGrupoCriterio` e `GrupoCriterio`.

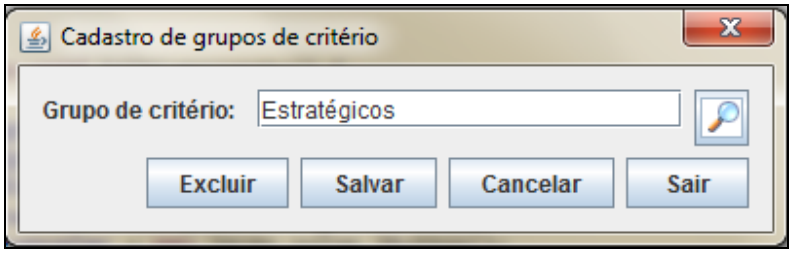


Figura 21 - Cadastro de grupo de critério

Por exemplo, para realizar a exclusão de um grupo de critério, é necessário antes realizar a pesquisa deste grupo de critério. Ao realizar a pesquisa e selecionar o grupo de critério que se deseja excluir, deve-se selecionar o botão `Excluir`. Ao selecionar o botão, é chamado o evento `btexcluirActionPerformed` do botão. O evento verifica se foi selecionado um grupo de critério para realizar a exclusão, e caso afirmativo, chama o método `excluirGrupoCriterio` passando como parâmetros o código do grupo de critério selecionado (Quadro 18).

```
private void btexcluirActionPerformed(java.awt.event.ActionEvent evt) {
       if (codigoGrupo != 0) {
              excluirGrupoCriterio(codigoGrupo);
       }else{
              JOptionPane.showMessageDialog(null,
                           "Para realizar a exclusão deve ser buscado um grupo
de critério antes!");
       }
    }
```

Quadro 18 - Evento do botão **`Excluir`**

O método `excluirGrupoCriterio` executa o método `excluirGrupoCriterio` da classe `GrupoCriterioDAO`, passando por parâmetro o código do grupo de critério informado na tela. O método `excluirGrupoCriterio` da classe `GrupoCriterioDAO`, realiza a conexão com o banco de dados e executa o comando de exclusão no banco de dados.

Caso a exclusão seja efetuada com sucesso, o método retorna o valor 0, caso ocorra algum erro durante a exclusão, o método retorna o valor -1. Conforme este retorno, o sistema

mostra a mensagem "Erro a tentar excluir o grupo de critério!" ou "Exclusão na tabela grupo de critério realizada com sucesso!" (Quadro 19).

```
private void excluirGrupoCriterio(int parmIdGrProjeto){
	int retorno = grupoCriterioDAO.excluirGrupoCriterio(parmIdGrProjeto);

	if(retorno == -1){
		JOptionPane.showMessageDialog(null,
				"Erro a tentar excluir o grupo de critério!");
	}else{
		valoresDefault();
		JOptionPane.showMessageDialog(null,
				"Exclusão na tabela grupo de critério realizada com
sucesso!");
	}
}
```

Quadro 19 - Método excluir grupo de critério

A classe GrupoCriterio possui todos os atributos do grupo de critério, juntamente com os métodos para atribuir valor e retornar o valor dos atributos. Essa classe é utilizada na tela de pesquisa dos grupos de critérios, da mesma maneira que foi apresentada a tela de pesquisa dos projetos (Figura 19) e pela rotina de comparação de projetos que será apresentada posteriormente.

3.3.2.3 Cadastrar critério

A terceira funcionalidade é cadastrar critério (caso de uso UC03) e é responsável pela criação, alteração e exclusão dos critérios. A funcionalidade é composta por três classes principais CriterioGUI, CriterioDAO e Criterio. A classe CriterioGUI é responsável pelo leiaute da tela de cadastro (Figura 22) e faz as chamadas das classes CriterioDAO, PesquisarCriterio e Criterio.

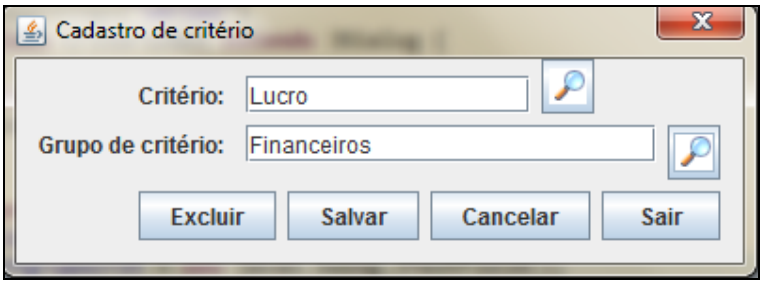


Figura 22 - Cadastro de critério

Por exemplo, para realizar a alteração de um critério, é necessário antes realizar a pesquisa do critério. Ao realizar a pesquisa e selecionar o critério que se deseja alterar, deve-se selecionar o botão Salvar. Ao selecionar o botão, é chamado o evento

btSalvarActionPerformed do botão. O evento verifica se o critério já possui código. Se ele possuir, chama o método alterarCriterio. Se não possui, chama o método adicionarCriterio (Quadro 20).

```
public void salvarDados(){
	if (codigoCriterio != 0) {
		alterarCriterio(codigoCriterio, tfCriterio.getText());
	}else{
		adicionarCriterio();
	}
}
```

Quadro 20 - Método do botão Salvar

No método alterarCriterio da classe CriterioGUI é chamado o método alterarCriterio da classe CriterioDAO passando como parâmetro o código do critério, a descrição do critério e o código do grupo de critério. O código do critério é buscado com base no critério selecionado na tela de pesquisa de critério. Já o código do grupo de critério pode ser buscando da mesma maneira do código de critério, já que todo critério possui uma associação com um grupo de critério, ou caso o Usuário alterar esse grupo de critério, irá buscar da tela de pesquisa do grupo de critério (Quadro 21).

```
private void alterarCriterio(int parmIdCriterio, String parmDsCriterio){

	if(codigoGrupo == 0){
		JOptionPane.showMessageDialog(null,
			"Deve ser informado o grupo de critério!");
	}else{
	int retorno = criterioDAO.alterarCriterio(parmIdCriterio, parmDsCriterio,
codigoGrupo);

		if(retorno == -1){
			JOptionPane.showMessageDialog(null,
					"Erro ao tentar atualizar o critério!");
		}else{
			JOptionPane.showMessageDialog(null,
			"Atualização na tabela critério realizado com sucesso!");
				valoresDefault();
		}
	}
}
```

Quadro 21 - Método alterar critério

Da mesma maneira que foi apresentada a inclusão de um novo projeto (Quadro 15), o método alterarCriterio da classe CriterioDAO faz a conexão com o banco de dados e atualiza os dados no banco de dados.

A classe Criterio possui todos os atributos do critério, juntamente com os métodos para atribuir valor e retorno valor dos atributos. Essa classe é utilizada na tela de pesquisa dos critérios, da mesma maneira que foi apresentada a tela de pesquisa dos projetos (Figura 19) e pela rotina de comparação de projetos que será apresentada a seguir.

3.3.2.4 Realizar comparação entre os projetos

A quarta funcionalidade é realizar comparação (caso de uso `UC07`) e é responsável por selecionar e priorizar os projetos. Essa funcionalidade possui várias etapas e por isso várias telas precisam ser apresentadas. Para que seja definida qual a ordem que essas telas serão apresentadas, foi utilizado o gerenciador *CardLayout*. Para utilizar esse gerenciador se faz necessário criar uma lista com o nome de todas as telas que serão apresentadas, e conforme o `Usuário` vai avançando nas etapas, o método `mostrarTela` é chamado e atualiza a tela que será exibida naquele momento.

```
private void mostrarTela(String parmNomeTela) {
	CardLayout card = (CardLayout) jPPrincipal.getLayout();
	card.show(jPPrincipal, parmNomeTela);
}
```

Quadro 22 - Método mostrar tela

A primeira tela apresentada na comparação é a de boas-vindas, explicando o objetivo da funcionalidade.

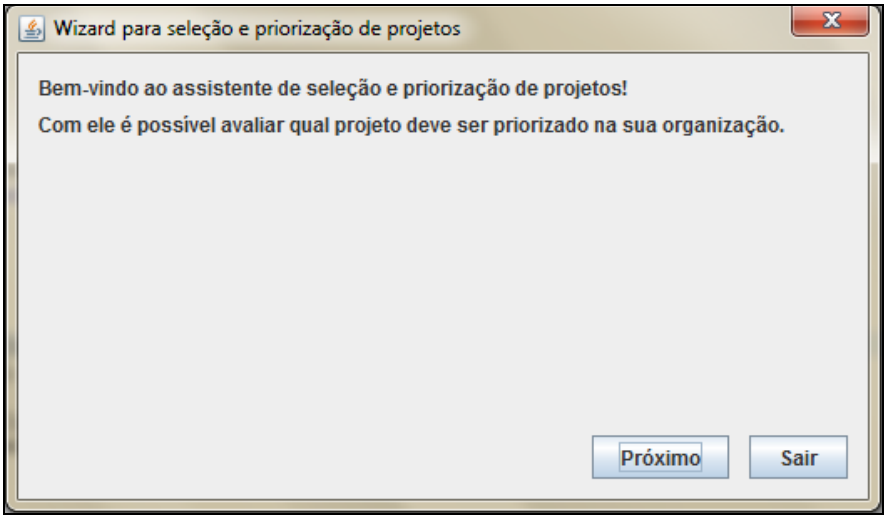


Figura 23 - Tela de boas-vindas

A próxima tela apresentada é a seleção dos projetos que serão priorizados. Para mostrar o nome de todos os projetos que foram cadastrados, foi criada uma tabela e a essa tabela foi associado um *TableModel*. Foi necessário criar o *TableModel* pois a tabela será criada dinamicamente, conforme a quantidade de projetos que estão cadastrados. Em seguida é chamado o método `buscaProjetos`, passando como parâmetro o *TableModel* criado.

O método `buscaProjetos` busca todos os projetos cadastrados chamando o método `buscaListaProjetos` da classe `PesquisarProjetoDAO`. Esse método retorna a lista com todos

os projetos cadastros. A aplicação percorre toda a lista e adiciona cada projeto no *TableModel* (Quadro 23).

```
private void buscaProjetos(ProjetoTableModel tableModel) throws Exception {

ArrayList<Projeto> listaTodosProjetos = listaProjetos.buscaListaProjetos();

      if(listaTodosProjetos.size() == 0){
            JOptionPane.showMessageDialog(null,
            "Nenhum projeto encontrado ou ocorreu problema durante a
consulta.");
      }else{
          for(int i = 0; i < listaTodosProjetos.size(); i++) {
             Projeto novoProjeto = new Projeto();
             novoProjeto.adicionaProjetoWizard(listaTodosProjetos.get(i)
                    .getNome(), novoProjeto);
             tableModel.adicionarProjeto(novoProjeto, i);
          }
       }
}
```

Quadro 23 - Buscar projetos

O nome da classe do *TableModel* é `ProjetoTableModel`. Essa classe possui uma lista com os projetos que devem ser exibidos. Através do método `adicionarProjeto`, o projeto é adicionado na lista e a tela é atualizada para mostrar esse projeto (Quadro 24).

```
public void adicionarProjeto(Projeto parmProjeto, int posicao) {
      listaProjetos.add(parmProjeto);
      fireTableRowsInserted(listaProjetos.size()-1, listaProjetos.size()-1);
}
```

Quadro 24 - Adicionar projeto na tela de comparação de projetos

Após todos os projetos serem adicionados na tela, o `Usuário` escolhe quais projetos deseja realizar a comparação.

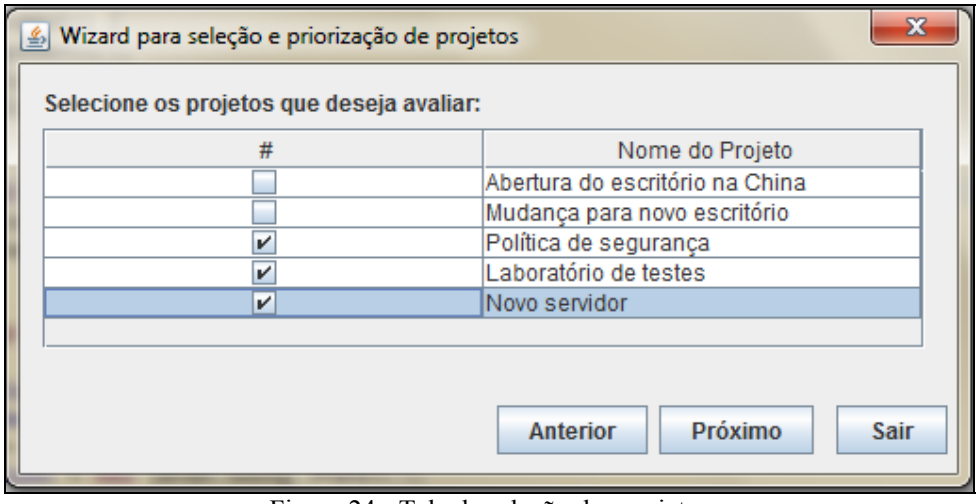


Figura 24 - Tela de seleção dos projetos

No caso do exemplo que será apresentado nessa seção, foram selecionados os projetos Política de segurança, Laboratório de testes e Novo servidor.

Sempre que um projeto for selecionado, o método `setValueAt` da classe `ProjetoTableModel` é chamado. Cada projeto possui um atributo com o nome `estaSelecionado`. Quando é feita a seleção do projeto, esse atributo recebe o valor verdadeiro. Caso a seleção não for feita, o atributo fica com o valor falso.

```
public void setValueAt(Object value, int row, int col) {
       if (row > -1 && row < listaProjetos.size()) {
          Projeto projeto = listaProjetos.get(row);
             switch (col) {
                case 0:
                if(projeto.getEstaSelecionado() == false){
                       projeto.setEstaSelecionado(true);
                       listaProjetosSelecionados.add(projeto);
                }else{
                       projeto.setEstaSelecionado(false);
                       listaProjetosSelecionados.remove(projeto);
                }
             }
       }
}
```

Quadro 25 - Método executado quando projeto é selecionado

Da mesma maneira que são adicionados todos os projetos cadastrados para seleção na tela de comparação de projetos, são mostrados posteriormente os grupos de critério (Figura 25) e critério (Figura 26) para seleção.

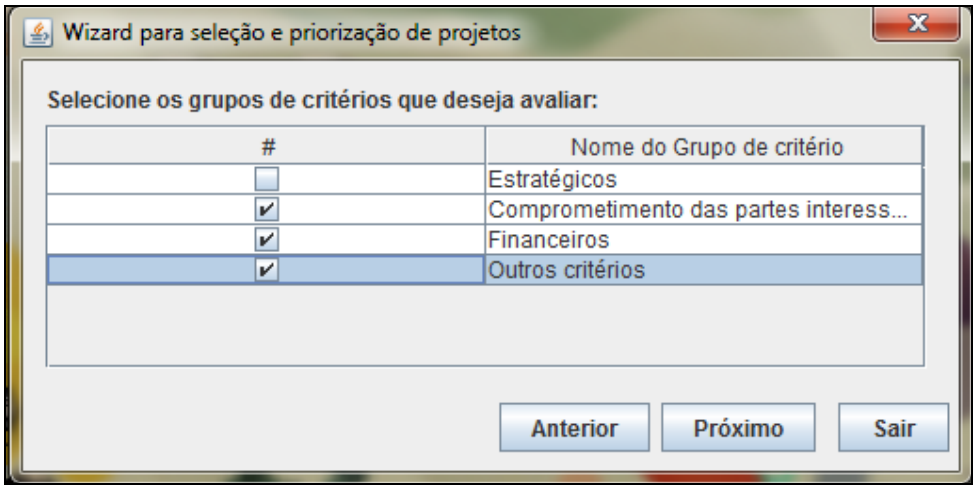


Figura 25 - Tela de seleção dos grupos de critérios

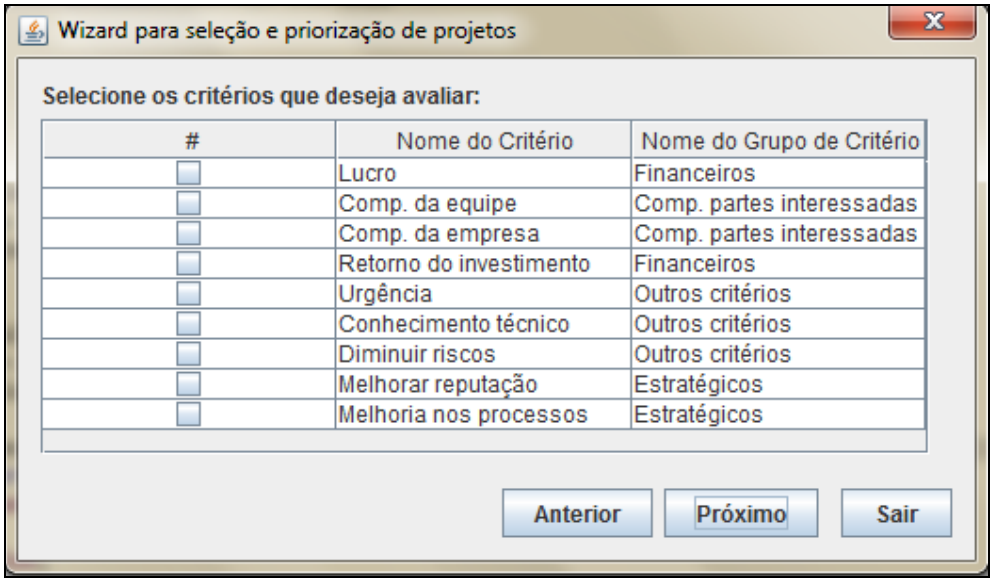


Figura 26 - Tela de seleção dos critérios

No exemplo apresentado foram selecionados os grupos de critérios Comprometimento das partes interessadas, Financeiros e Outros critérios. Já para os critérios foram selecionados Comprometimento da equipe e Comprometimento da empresa, que estão associados ao grupo de critério Comprometimento das partes interessadas. Os critérios Retorno do investimento e Lucro que estão associados ao grupo de critério Financeiros. E os critérios Diminuir riscos (ameaças) para a empresa, Urgência e Conhecimento técnico interno que estão associados ao grupo de critério Outros critérios.

Após selecionar os projetos que serão comparados, juntamente com os grupos de critérios e critérios que serão considerados, é necessário primeiramente informar o peso entre cada grupo de critério selecionado, que no exemplo são comprometimento das partes interessadas, financeiros e outros critérios (Figura 27).

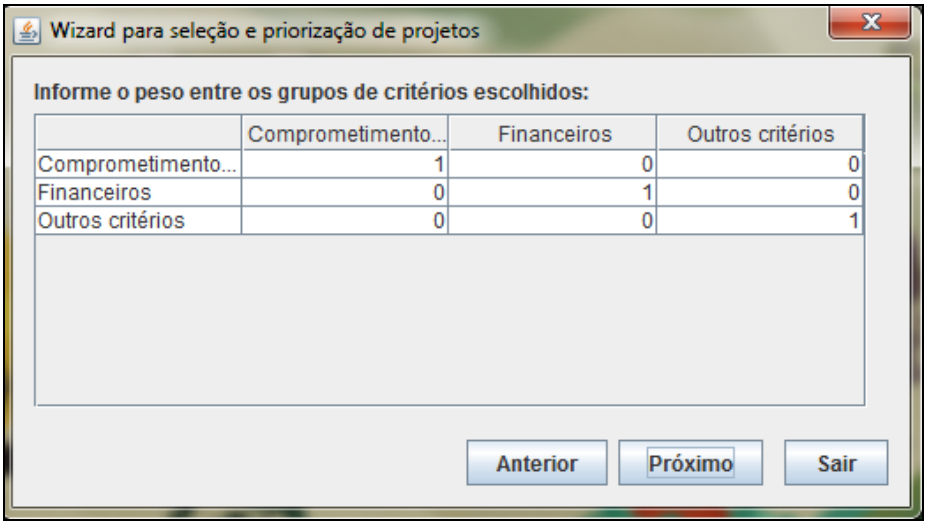


Figura 27 - Tela para informar peso entre os grupos de critérios

Para realizar a montagem dessa tela, assim como na seleção de projetos, foi necessário criar um *TableModel* específico para isso. Na criação do *TableModel* é passado como parâmetro a lista dos grupos de critérios selecionados. Também na criação do *TableModel* é criada uma matriz com o tamanho igual a quantidade de grupos de critérios selecionados, para serem armazenados os pesos informados (Quadro 26).

```
public PesoGrupoTableModel(List<Base> list) {
       this.listaGruposSelecionados = new ArrayList<Base>();
       this.listaGruposSelecionados = list;
       matrizValores = new double [list.size()-1][list.size()-1];
       populaMatriz();
}
```

Quadro 26 - Criação do *TableModel* para peso dos grupos de critérios

Essa matriz já é criada com o valor 1 (um) para as intersecções dos grupos de critérios e 0 (zero) para o restante dos campos (Quadro 27).

```
public void populaMatriz(){
      for (int i = 0; i < matrizValores.length; i++){
            for (int j = 0; j < matrizValores.length; j++){
                  if(i == j){
                        matrizValores[i][j] = 1;
                  }else{
                        matrizValores[i][j] = 0;
                  }
            }
      }
}
```

Quadro 27 - Criação da matriz para armazenar os pesos informados

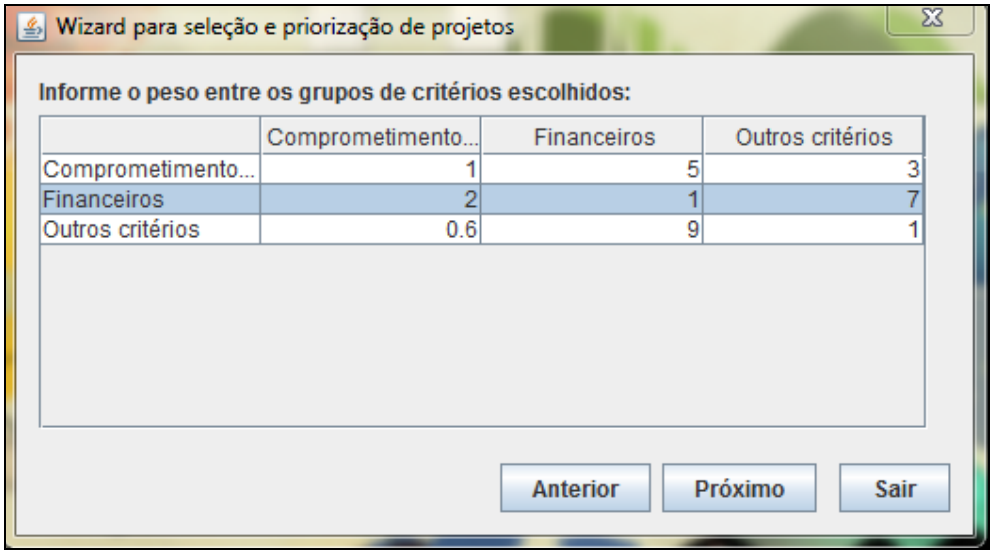
Após a criação do *TableModel* é executado o método `getValueAt` que irá colocar o nome do grupo ou o valor do peso para cada célula da matriz da tela (Quadro 28).

```
public Object getValueAt(int rowIndex, int columnIndex) {
      if (rowIndex > -1 && rowIndex < listaGruposSelecionados.size()-1) {
            GrupoCriterio grupo = (GrupoCriterio)
listaGruposSelecionados.get(rowIndex+1);
            switch (columnIndex) {
            case 0:
                  return grupo.getNomeGrupo();
            default:
                  return matrizValores[rowIndex][columnIndex-1];
            }
      }
            return null;
}
```

Quadro 28 - Valores padrão da tela de peso

Sempre que for informado um valor para alguma célula da matriz, é chamado o método `setValueAt`. Esse método busca o valor informado na tela e coloca na matriz criada no *TableModel*.

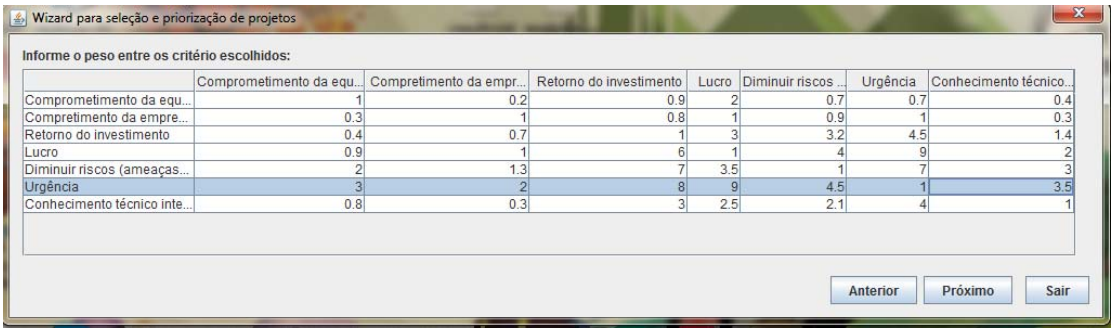
Esse mesmo procedimento é realizado para os critérios selecionados. No caso do exemplo apresentado, foram informados os pesos entre os grupos de critério conforme pode ser visto na Figura 28.



	Comprometimento...	Financeiros	Outros critérios
Comprometimento...	1	5	3
Financeiros	2	1	7
Outros critérios	0.6	9	1

Figura 28 - Peso informado entre os grupos de critérios selecionados

Já entre os critérios foram informados os pesos conforme pode ser visto na Figura 29.



	Comprometimento da equ...	Compretimento da empr...	Retorno do investimento	Lucro	Diminuir riscos ...	Urgência	Conhecimento técnico...
Comprometimento da equ...	1	0.2	0.9	2	0.7	0.7	0.4
Compretimento da empre...	0.3	1	0.8	1	0.9	1	0.3
Retorno do investimento	0.4	0.7	1	3	3.2	4.5	1.4
Lucro	0.9	1	6	1	4	9	2
Diminuir riscos (ameaças...	2	1.3	7	3.5	1	7	3
Urgência	3	2	8	9	4.5	1	3.5
Conhecimento técnico inte...	0.8	0.3	3	2.5	2.1	4	1

Figura 29 - Peso informado entre os critérios selecionados

Após informar os pesos entre cada grupo de critério e critério é realizado o cálculo para definir o peso geral de cada grupo de critério e critério. Para realizar os cálculos, foi criada a classe `MatrizCN`. A classe `WizardGUI` chama vários métodos da classe `MatrizCN` para chegar no resultado final (Quadro 29).

```
private void calculaPesoGrupoECriterio() {
	matrizComparativaGrupo = new MatrizCN();
	matrizComparativaGrupo.defineMatrizComparativa((ArrayList<Base>)
tableModelGrupo.getListaGruposSelecionados());
	matrizComparativaGrupo.definePesos(tableModelPesoGrupo.getVetorValores()
);
	matrizComparativaGrupo.calculaTotalMatriz();
	matrizComparativaGrupo.calculaMediaPesos();

	matrizComparativaGrupo.calculaVetorEigen();
}
```

Quadro 29 - Cálculo do peso geral dos grupos de critérios

Primeiramente através do método `defineMatrizComparativa`, são definidos os tamanhos das matrizes que serão utilizadas nos cálculos. Essas matrizes terão o tamanho conforme a quantidade de grupos de critérios selecionados. Em seguida, são definidos os pesos da matriz conforme foi informado pelo `Usuário` (Figura 27). O primeiro cálculo a ser realizado é o total da matriz. Para a realização desse cálculo são somados os pesos informados de cada coluna e armazenada em uma nova matriz para ser utilizada posteriormente (Quadro 30).

```
public void calculaTotalMatriz(){
	for (int l = 0; l < qtd; l++){
		float temp = 0;
		for (int c = 0; c < qtd; c++){
			temp += this.matriz[c][l];
		}
		this.matrizTotal[l] = temp;
	}
}
```

Quadro 30 - Cálculo do total da matriz

No caso do exemplo, ao realizar o cálculo do total da matriz para os grupos de critérios, chega-se no resultado apresentado na Tabela 6. Durante a explicação dos cálculos realizados no exemplo, serão apresentadas várias tabelas com o propósito de auxiliar no entendimento dos cálculos realizados internamento pela aplicação.

Tabela 6 - Valor total da matriz entre os grupos de critérios

	Comprometimento	Financeiros	Outros critérios
Comprometimento	1	5	3
Financeiros	2	1	7
Outros critérios	0,6	9	1
Total	**3,6**	**15**	**11**

Em seguida é calculada a média de cada célula da matriz. Para a realização desse cálculo é buscado o peso informado para cada célula da matriz dos grupos de critérios e dividido pelo total da coluna.

```
public void calculaMediaPesos(){
	for (int l = 0; l < qtd; l++){
		for (int c = 0; c < qtd; c++){
			this.matrizMediaPesos[l][c] = matriz[l][c] /
			this.matrizTotal[c];
		}
	}
}
```

Quadro 31 - Cálculo da média da matriz

No caso do exemplo, ao realizar o cálculo da média da matriz para os grupos de critérios, chega-se ao resultado apresentado na Tabela 7.

Tabela 7 - Cálculo da média das matrizes entre os grupos de critérios

Comprometimento	1/3,6 = 0,278	0,333	0,273
Financeiros	2/3,6 = 0,556	0,067	0,634
Outros critérios	0,6/3,6 = 0,167	0,6	0,091

Por fim é realizado o cálculo do vetor de Eigen. O cálculo é feito através da soma de todas as células da mesma linha divida pela quantidade de elementos que a linha possui (Quadro 32).

```
public void calculaVetorEigen(){
	for (int l = 0; l < qtd; l++){
		float temp = 0;
		for (int c = 0; c < qtd; c++){
			temp += this.matrizMediaPesos[l][c];
		}

		this.vetorEigen[l] = Util.arredondar((temp/qtd), 4, 1);
	}
}
```

Quadro 32 - Cálculo do Vetor de Eigen

O resultado de cada linha representa o valor do peso geral de um grupo de critério. Para o exemplo que está sendo apresentado, ao realizar o cálculo do vetor de Eigen, chega-se ao valor apresentado na Tabela 8.

Tabela 8 - Cálculo do vetor de Eigen para os grupos de critérios

Comprometimento	[0,278 + 0,333 + 0,273] / 3 = 0,294 (29,4%)
Financeiros	[0,556 + 0,067 + 0,634] / 3 = 0,419 (41,9%)
Outros critérios	[0,167 + 0,6 + 0,091] / 3 = 0,286 (28,6%)

Esse mesmo procedimento é feito para os critérios selecionados e ao final tem-se o valor geral do peso de cada grupo de critério e critério selecionado, conforme pode ser visto na Figura 30.

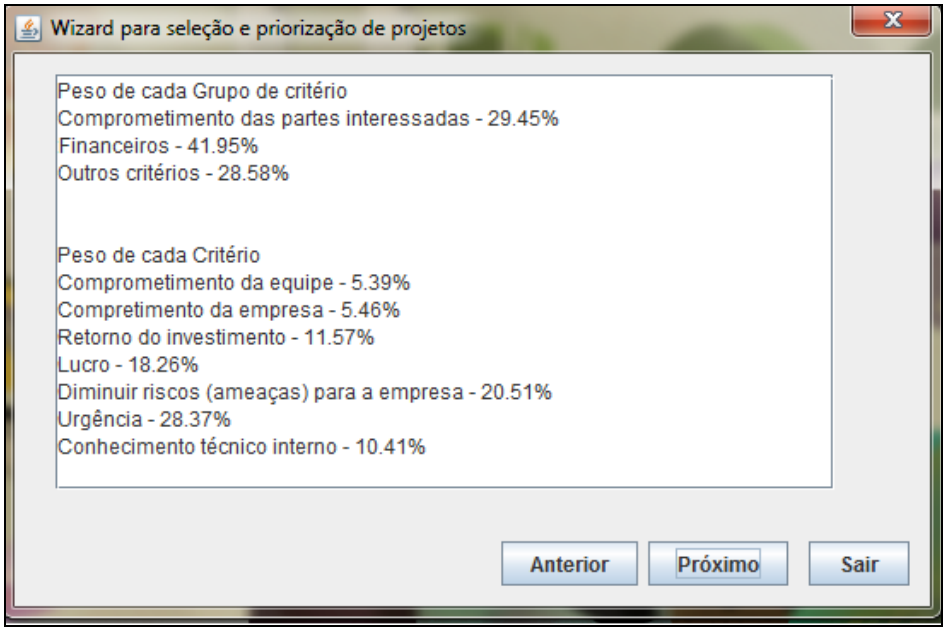


Figura 30 - Peso geral entre cada grupo e critério selecionado

Todos os pesos são informados pelo usuário da aplicação. Para garantir que os valores estão corretos, é realizado o cálculo do índice de consistência. Caso o valor seja considerado inconsistente, é gerada uma mensagem de alerta pela aplicação (Figura 31).

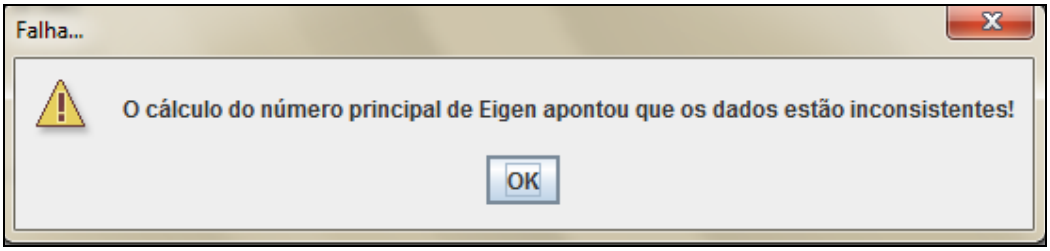


Figura 31 - Mensagem do índice de consistência

A próxima etapa é definir o peso entre os projetos para cada grupo de critérios e critérios informados. Primeiramente são apresentados todos os grupos de critérios e projetos selecionados e é informado o peso para esses grupos. Para saber quantos grupos de critérios foram selecionados e saber a ordem para apresentar esses grupos de critérios, foi criada a variável `contTotalGrupo`, que possui o número total de grupos selecionados. A primeira vez que o método `exibePesoProjetoPorGrupo` for chamado, essa variável irá possuir o número total de grupos de critérios selecionados. Ao final desse método, essa variável é decrementada e chama o método novamente até que todos os grupos de critérios sejam mostrados na tela (Quadro 33).

```
private void exibePesoProjetoPorGrupo() throws Exception {
	String nomeGrupo = tableModelGrupo.getListaGruposSelecionados()
		.get(tableModelGrupo.getListaGruposSelecionados().size()
			- contTotalGrupo).getNome();

	jLPesoProjeto.setText("Informe os pesos entre os projetos considerando o
grupo de critério: " + nomeGrupo);

	tableModelPesoProjeto = new PesoProjetoTableModel(
	tableModelProjeto.getListaProjetosSelecionados());

	jTPesoProjeto.setModel(tableModelPesoProjeto);
	listaTableModelPesoProjetoGrupo[tableModelGrupo
		.getListaGruposSelecionados().size() - contTotalGrupo] =
		tableModelPesoProjeto;
		contTotalGrupo--;
}
```

Quadro 33 - Exibir peso entre os projetos para cada grupo de critério

A tela que é apresentada possui uma matriz com os projetos selecionados e na sua descrição, possui a qual grupo de critério deve-se informar os pesos (Figura 32).

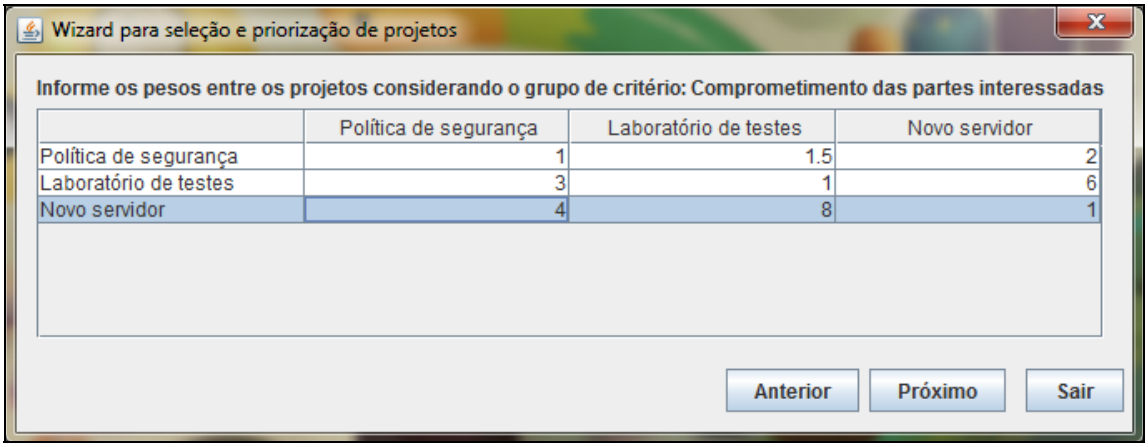


Figura 32 - Informar peso entre os grupos de critérios

Para o exemplo demonstrado, os pesos informados entre cada projeto precisam ser executados entre os grupos de critérios selecionados Comprometimento, Financeiros e Outros critérios. Para o grupo de critério Comprometimento foram informados os pesos apresentados na Tabela 9.

Tabela 9 - Pesos entre os projetos para o grupo de critério Comprometimento

Comprometimento	Política de segurança	Laboratório de testes	Novo servidor
Política de segurança	1	1.5	2
Laboratório de testes	3	1	6
Novo servidor	4	8	1

Para o grupo de critério Financeiros foram informados os pesos apresentados na Tabela 10.

Tabela 10 - Pesos entre os projetos para o grupo de critério Financeiros

Financeiros	Política de segurança	Laboratório de testes	Novo servidor
Política de segurança	1	5	9
Laboratório de testes	4	1	11
Novo servidor	3	8	1

Para o grupo de critério Outros critérios foram informados os pesos apresentados na Tabela 11.

Tabela 11 - Pesos entre os projetos para o grupo de critério Outros critérios

Financeiros	Política de segurança	Laboratório de testes	Novo servidor
Política de segurança	1	3,5	4
Laboratório de testes	2	1	9
Novo servidor	3	7	1

O mesmo procedimento é executado para todos os critérios selecionados. No exemplo descrito são informados os pesos entre os projetos para os critérios Comprometimento da equipe, Comprometimento da empresa, Retorno do investimento, Lucro, Diminuir riscos (ameaças) para a empresa, Urgência e Conhecimento técnico interno.

Para o critério Comprometimento da equipe foram informados os pesos apresentados na Tabela 12.

Tabela 12 - Pesos entre os projetos para o critério Comprometimento da equipe

Comprometimento da equipe	Política de segurança	Laboratório de testes	Novo servidor
Política de segurança	1	3,5	4
Laboratório de testes	2	1	9
Novo servidor	3	7	1

O `Usuário` irá informar o peso entre todos os critérios citados acima, porém neste exemplo são mostrados apenas os pesos informados para o critério Comprometimento da equipe.

Ao final, são realizados os cálculos dos projetos com base nos pesos informados a cada grupo de critério e critério selecionado. São executados os mesmo métodos já descritos no Quadro 29 para cada matriz com os pesos dos grupos e critérios. O resultado do peso geral de cada projeto por grupo de critério é apresentado na Tabela 13.

Tabela 13 - Peso geral entre os projetos para cada grupo de critério

	Comprometimento	Financeiros	Outros critérios
Novo servidor	0,1632	0,3035	0,2522
Política de segurança	0,3789	0,365	0,3543
Laboratório de testes	0,4576	0,3312	0,3933

O resultado do peso geral de cada projeto dos critérios dos grupos de critérios Comprometimento e Financeiros é apresentado na Tabela 14.

Tabela 14 - Peso geral entre os critérios Comprometimento e Financeiros

	Comprometimento da equipe	Comprometimento da empresa	Retorno do investimento	Lucro
Novo servidor	0,2522	0,3051	0,3103	0,3733
Política de segurança	0,3543	0,425	0,2511	0,2606
Laboratório de testes	0,3933	0,2697	0,4384	0,366

O resultado do peso geral de cada projeto dos critérios de Outros critérios é apresentado na Tabela 15.

Tabela 15 - Peso geral entre os critérios Outro critérios

	Diminuir riscos (ameaças) para a empresa	Urgência	Conhecimento técnico interno
Novo servidor	0,4058	0,3644	0,3099
Política de segurança	0,2901	0,2679	0,3899
Laboratório de testes	0,3039	0,3679	0,300

Para chegar a priorização de cada projeto, é ainda executado mais um cálculo. Deve-se somar o valor do peso geral de todos os grupos de critérios do mesmo projeto, então se obtém

o peso geral do projeto entre os grupos de critério (Quadro 34). O mesmo procedimento deve ser feito entre todos os critérios para ter o peso geral do projeto entre os critérios.

```
for (int x = 0; x < qtdProjetos; x++) {
	vetorFinalGrupo[(i * qtdProjetos) + x] = vetorPesoGrupo[i]
		* matrizComparativaPesoProjetoGrupo.getVetorEigen()[x];
}

int qtdGrupos = tableModelGrupo.getListaGruposSelecionados().size();

for (int p = 0; p < qtdProjetos; p++) {
	double total = 0;
	for (int g = 0; g < qtdGrupos; g++) {
		total = total + vetorFinalGrupo[p + (qtdProjetos * g)];
}

resultadoFinal += "Projeto: " +
tableModelProjeto.getListaProjetosSelecionados().get(p).getNome() + " " +
"Total: " + total + "\n";
```

Quadro 34 - Soma dos pesos de todos os grupos de critério do mesmo projeto

No caso do exemplo demonstrado, somando o peso de todos os grupos de critérios e critérios do mesmo projeto, chega-se ao valor apresentado na Tabela 16.

Tabela 16 - Média entre os grupos de critério e critérios

	Peso geral entre todos os grupos de critérios	Peso geral entre todos os critérios	Média entre os pesos
Novo servidor	0,2475	0,3532	0,3004
Política de segurança	0,3660	0,2950	0,3305
Laboratório de testes	0,3862	0,3513	0,7375

Realizando uma média entre o peso geral de todos os grupos de critério e os critérios selecionados, obtêm-se os valores de 30,04% de prioridade para a compra do novo servidor, 33,05% de prioridade para a aplicação de uma nova política de segurança e 73,75% de prioridade para a montagem de um novo laboratório de testes.

Após realizar a priorização dos projetos, é possível salvar um relatório em *Portable Document Format* (PDF) com a lista dos projetos ordenados por sua priorização, ou um relatório com a lista dos grupos de critério e critérios com o peso geral deles e exportar toda a comparação para um arquivo *eXtensible Markup Language* (XML) para ser utilizado posteriormente (Figura 33).

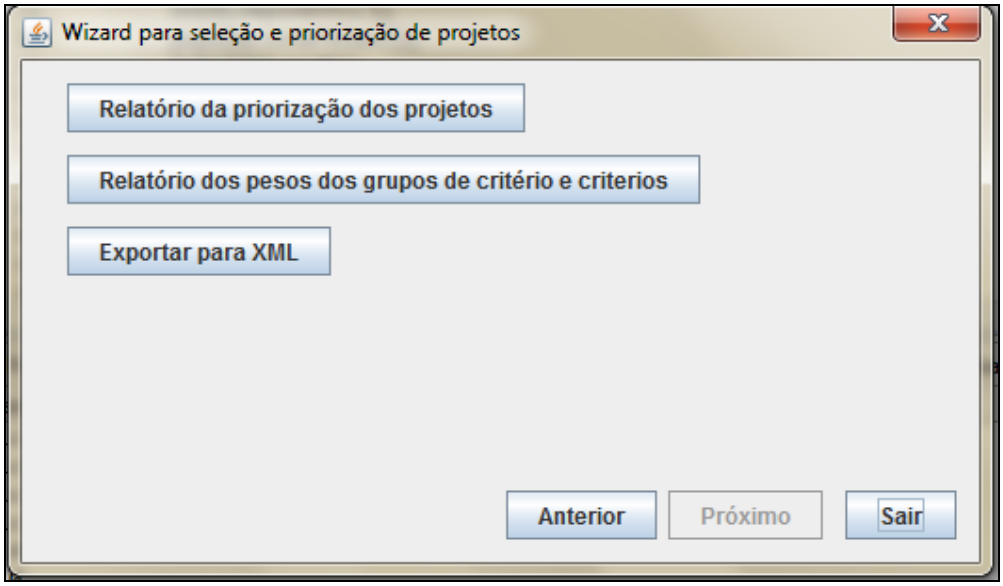


Figura 33 - Seleção de relatório e exportar para XML

3.3.2.4.1 Relatório de projetos

O relatório de projetos (`UC08`) é responsável por gerar um relatório PDF com todos os projetos selecionados na comparação, ordenados por sua priorização. Para viabilizar esse relatório, foram implementadas duas classes principais `RelatorioProjetoGUI` e `RelatorioProjeto`. A classe `RelatorioProjetoGUI` é responsável pelo leiaute da tela de visualização dos projetos com o seu peso final (Figura 34) e faz a chamada da classe `RelatorioProjeto`.

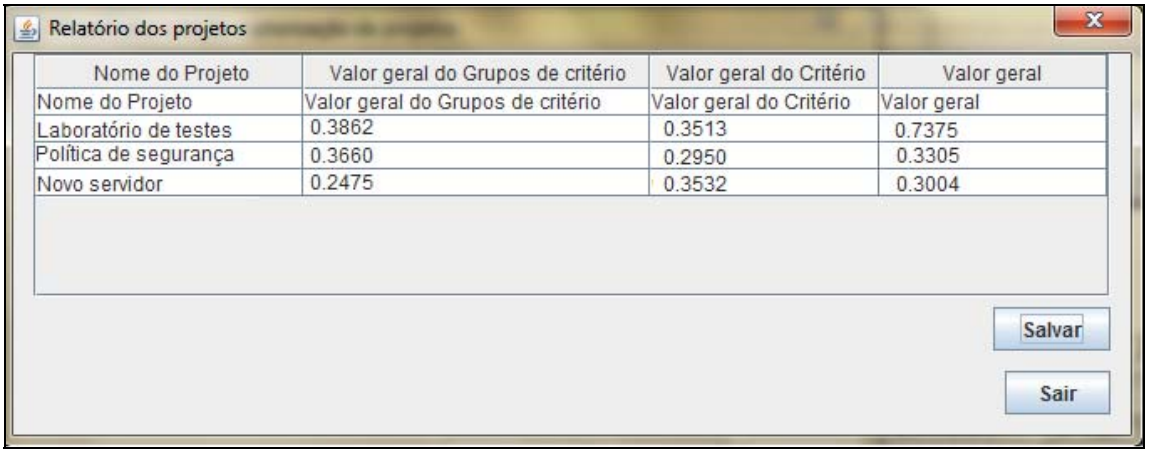


Figura 34 - Relatório dos projetos

Para possibilitar a geração do relatório, são necessárias as informações do nome do projeto, peso geral do projeto por grupo de critério, peso geral do projeto por critério e a

média dos pesos. Para conseguir os valores do peso geral do projeto por grupo de critério e por critério, foram criados os métodos `adicionarGrupo` e `adicionarCriterio` na classe `RelatorioProjeto`. Sempre que a classe `WizardGUI` realizar o cálculo do peso do grupos de critério e critério (explicado na seção 3.3.2.4), serão chamados esses dois métodos, conforme é demonstrado no Quadro 35.

```
resultadoFinal += "Projeto: " +
tableModelProjeto.getListaProjetosSelecionados().get(p)
       .getNome() + " " + "Total: " + total + "\n";

relatorioProjeto.adicionarGrupo(tableModelProjeto.getListaProjetosSe
lecionados().get(p) .getNome(), total, p);
```

Quadro 35 - Método para adicionar o peso geral do grupo de critério

O método `adicionarGrupo` recebe como parâmetro o nome do projeto, o valor geral do peso do projeto para o grupo de critério que está realizando o cálculo e a posição que deve ser adicionado essa informação na matriz.

O método `adicionarGrupo` por sua vez adiciona na matriz `matrizGrupos` na coluna zero (0) o nome do projeto e na coluna um (1) o valor do peso geral do grupo de critério para o projeto (Quadro 36).

```
public void adicionarGrupo(String nomeProjeto, double vlr, int
posicao) {
       matrizGrupos[posicao][0] = nomeProjeto;
       matrizGrupos[posicao][1] = String.valueOf(vlr);
}
```

Quadro 36 - Método `adicionarGrupo`

A matriz `matrizGrupos` foi criada no construtor da classe `RelatorioProjeto` e possui o tamanho igual a quantidade de grupos de critérios selecionados na comparação. O mesmo procedimento explicado nos quadros Quadro **35** e Quadro **36**, são executados para os pesos dos critérios. Assim quando o usuário selecionar a opção para gerar o relatório com os projetos, a classe `RelatorioProjeto` irá possuir uma matriz com o nome dos projetos e o peso geral do projeto por grupo de critério e outra matriz com o nome dos projetos e o peso geral do projeto por critério.

Ao selecionar a opção para gerar o relatório com os projetos, é chamado o método `criarTabela` da classe `RelatorioProjeto`. Esse método cria a tabela que será utilizada pelo relatório com o nome das colunas e chama o método `montaDados`. O método `montaDados` é responsável por percorrer as duas matrizes que foram preenchidas durante o cálculo da priorização (Quadro 35) e adicionar na tabela utilizada pelo relatório (Quadro 37).

```
public void montaDados(DefaultTableModel modelo) throws Exception{
       PesquisarAcompanhamentoDAO acompanhamento = new
PesquisarAcompanhamentoDAO();
       ArrayList<Acompanhamento> listaAcompanhamentos =
acompanhamento.buscaListaTodosAcompanhamentos();
       for(int i = 0; i < listaAcompanhamentos.size(); i++){
              String codProjeto =
String.valueOf(listaAcompanhamentos.get(i).getCodigoProjeto());
              String dsAcompanhamento =
listaAcompanhamentos.get(i).getDsAcompanhamento();
              adicionaTable(modelo, codProjeto, dsAcompanhamento);
       }
}
```

Quadro 37 - Montar os dados para a tabela que será utilizada no relatório

Após a montagem da tabela é possível salvar o relatório com a priorização dos projetos. Para possibilitar a geração do arquivo PDF, foi utilizada a API *IText*. Essa API é gratuita e *open source*.

Ao selecionar a opção para salvar o relatório de projetos, é chamado o método `createPDF` da classe `geraPDF`. Esse método recebe a tabela e gera o arquivo PDF no diretório do projeto Quadro 38.

```
public static void createPdf(JTable parmTable, String nomeArquivo) {
       Document document = new Document();
                try {
                   PdfWriter writer;
                     writer = PdfWriter.getInstance(document,
                   new FileOutputStream(nomeArquivo + ".pdf"));

                   document.open();
                   PdfContentByte cb = writer.getDirectContent();
                   PdfTemplate tp = cb.createTemplate(500, 500);
                   Graphics2D g2;
                     g2 = tp.createGraphics(500, 500);
                   parmTable.print(g2);
                   g2.dispose();
                   cb.addTemplate(tp, 30, 300);
                   } catch (Exception e) {
                   System.err.println(e.getMessage());
       }
       document.close();
}
```

Quadro 38 - Método para gerar o arquivo PDF

Após executar o método `createPDF` será salvo um arquivo PDF com uma lista dos projetos selecionados juntamente com o peso da priorização dos projetos (Figura 35).

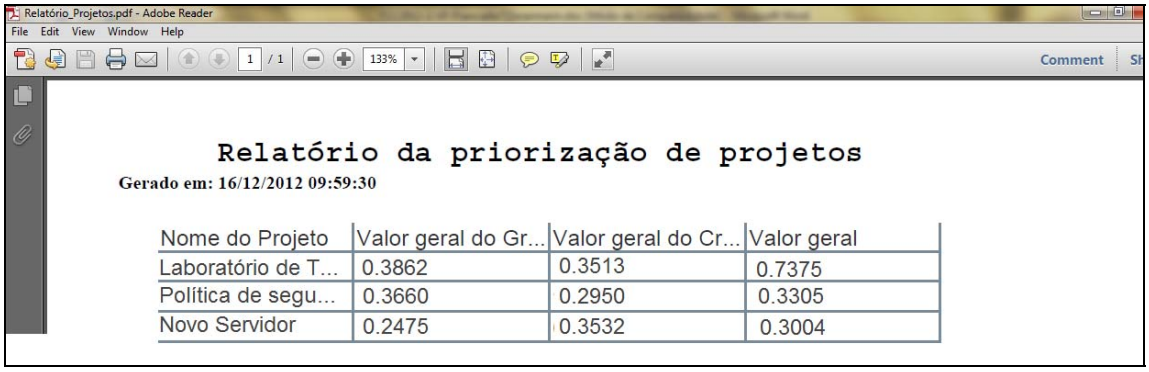


Figura 35 - Arquivo PDF com o relatório dos projetos

3.3.2.4.2 Relatório dos pesos dos grupos de critérios e critérios

O relatório de pesos dos grupos de critério e critérios (`UC09`) funciona da mesma maneira que o relatório de projetos, explicado na seção anterior. A diferença é que o pesos calculados para os grupos de critério e critérios são passados para os métodos `adicionarGrupo` e `adicionarCriterio` da classe `RelatorioPesos` durante o cálculo do peso geral (Quadro 39).

```
String mensagem = "Peso de cada "+ parmTipo + " \n";
	for (int l = 0; l < qtd; l++){
	mensagem += parmList.get(l).nome +
		" - " + Util.arredondar(vetorEigen[l] * 100,2,1) + "% \n" ;

if(parmTipo == "Grupo de critério"){
	parmRelatorioPesos.adicionarGrupo(parmList.get(l).nome, vetorEigen[l], l);
}else{
	parmRelatorioPesos.adicionarCriterio(parmList.get(l).nome, vetorEigen[l],
l);
}
```

Quadro 39 - Adicionar peso dos grupos e critérios

3.3.2.4.3 Exportar para XML

Para realizar a exportação das informações utilizadas durante a comparação dos projetos, foi criada a classe `Serializacao`. Essa classe possui listas com as informações dos projetos, grupos de critérios e critérios selecionados. Os pesos informados entre os grupos de critério e critérios e os pesos informados entre os projetos para cada grupo de critério e critério selecionado (Quadro 40).

```
private HashMap<Integer, Boolean> listaProjetosSelecionados;
private HashMap<Integer, Boolean> listaGruposCriteriosSelecionados;
private HashMap<Integer, Boolean> listaCriteriosSelecionados;
private HashMap<String, Double> listaPesoGrupos;
private HashMap<String, Double> listaPesoCriterios;
private HashMap<String, Double> listaPesoProjetoGrupo;
private HashMap<String, Double> listaPesoProjetoCriterios;
```

Quadro 40 - Lista com as informações da comparação de projetos

Ao selecionar a opção para exportar as informações para um arquivo XML, é criada uma instância desta classe e preenche-se as listas com os valores informados. Primeiramente são preenchidas as listas com os projetos, grupos de critério e critérios selecionados. Para isto, é buscada a lista com os projetos selecionados do *Table Model* e chamado o método `setListaProjetosSelecionados` passando como parâmetro o código do projeto (Quadro 41). Esse mesmo procedimento é realizado para as listas dos grupos de critério e critérios selecionados.

```
Serializacao serializacao = new Serializacao();
// Exporta os projetos selecionados
ArrayList<Base> listaProjetos = tableModelProjeto.getListaProjetosSelecionados();
for (int i = 0; i < listaProjetos.size(); i++) {
      serializacao.setListaProjetosSelecionados(listaProjetos.get(i).codigo);
}
```

Quadro 41 - Método para exportar os projetos selecionados

Em seguida, são exportados os pesos informados entre os grupos de critério e critérios. Para identificar de qual célula o peso está se referindo, foi criada uma chave no *HashMap* com o nome do grupo de critério da linha concatenado com o nome do grupo de critério da coluna (Quadro 42). Assim quando este XML for importado, será possível identificar em qual posição deve-se colocar o peso. Esse mesmo procedimento é realizado para exportar os pesos entre os critérios selecionados.

```
// Exporta os pesos entre os grupos de critérios
double[][] valoresGrupos = tableModelPesoGrupo.getVetorValores();
for (int i = 0; i < valoresGrupos.length; i++) {
      for (int j = 0; j < valoresGrupos.length; j++) {
       String nomeGrupo =
tableModelGrupo.getListaGruposSelecionados().get(i).getNome()
      + tableModelGrupo.getListaGruposSelecionados().get(j).getNome();
      double valorPeso = valoresGrupos[i][j];
      serializacao.setListaPesoGrupos(nomeGrupo, valorPeso);
      }
}
```

Quadro 42 - Método para exportar os pesos entre os grupos de critérios

Por fim, são exportados os pesos informados entre os projetos para cada grupo de critério e critério selecionado. Para identificar de qual célula o peso está referindo-se, foi criada uma chave no *HashMap* com o nome do grupo de critério concatenado com o nome do projeto da linha concatenado com o nome do projeto da coluna (Quadro 43). Esse mesmo procedimento é realizado para exportar os pesos entre os projetos para os critérios

selecionados.

```
// Exporta os pesos entre os projetos por grupo de critério
for(int i = 0; i < tableModelGrupo.getListaGruposSelecionados().size(); i++){
      for(int j = 0; j <
tableModelProjeto.getListaProjetosSelecionados().size(); j++){
            for(int x = 0; x <
tableModelProjeto.getListaProjetosSelecionados().size(); x++){

            String nomeProjeto =
      tableModelGrupo.getListaGruposSelecionados().get(i).getNome()
       + tableModelProjeto.getListaProjetosSelecionados().get(j).getNome()
       + tableModelProjeto.getListaProjetosSelecionados().get(x).getNome();

double valorPeso = listaTableModelPesoProjetoGrupo[i].getVetorValores()[j][x];

      serializacao.setListaPesoProjetoGrupo(nomeProjeto, valorPeso);
            }
      }
}
```

Quadro 43 - Método para exportar os pesos entre os projetos

Ao final, é chamado o método `gerarXml` da classe `Serializacao`. Esse método utiliza a biblioteca *XStream*. Essa biblioteca é gratuita e é utilizada para serializar objetos para XML. Através do método `toXML` da biblioteca, o método serializa todas as listas preenchidas conforme a explicação acima e serializa para exportar pro arquivo XML (Quadro 44).

```
public static void gerarXml(Serializacao serializacao, String local) {
      // Criamos o objeto que fará o trabalho de gerar o xml
      XStream xStream = new XStream(new DomDriver());
      // O método toXML() transforma nosso objeto em um padrão de saída
no formato XML.
      String documento = xStream.toXML(serializacao);
      // A chamada ao método salvaArquivo passando por parâmetro
      // a String com o XML gerado e o nome do arquivo
      salvarArquivo(documento, local+ "gerarXml.xml");
}
```

Quadro 44 - Método para gerar o arquivo XML

O arquivo XML é gerado com todas as informações que foram informadas durante a comparação dos projetos (Figura 36).

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?>
<utils.Serializacao>
  <listaProjetosSelecionados>
    <entry>
      <int>18</int>
      <boolean>true</boolean>
    </entry>
    <entry>
      <int>20</int>
      <boolean>true</boolean>
    </entry>
  </listaProjetosSelecionados>
  <listaGruposCriteriosSelecionados>
    <entry>
      <int>17</int>
      <boolean>true</boolean>
    </entry>
    <entry>
      <int>16</int>
      <boolean>true</boolean>
    </entry>
  </listaGruposCriteriosSelecionados>
  <listaCriteriosSelecionados>
    <entry>
      <int>16</int>
      <boolean>true</boolean>
    </entry>
    <entry>
      <int>19</int>
      <boolean>true</boolean>
    </entry>
    <entry>
      <int>14</int>
      <boolean>true</boolean>
    </entry>
  </listaCriteriosSelecionados>
```

Figura 36 - Exemplo do arquivo XML gerado

3.3.2.5 Relatório de acompanhamento de projeto

A última funcionalidade é o relatório de acompanhamento dos projetos (`UC10`) e é responsável por gerar um relatório PDF com todos os registros de acompanhamentos cadastrados em todos os projetos. A funcionalidade é composta por três classes principais, a `RelatorioAcompanhamentoGUI`, a `RelatorioAcompanhamentoDAO` e a `RelatorioAcompanhamento`. A classe `RelatorioAcompanhamentoGUI` é responsável pelo leiaute da tela de visualização dos acompahamentos de projeto (Figura 37) e faz as chamadas das classes `RelatorioAcompanhamentoDAO` e `RelatorioAcompanhamento`.

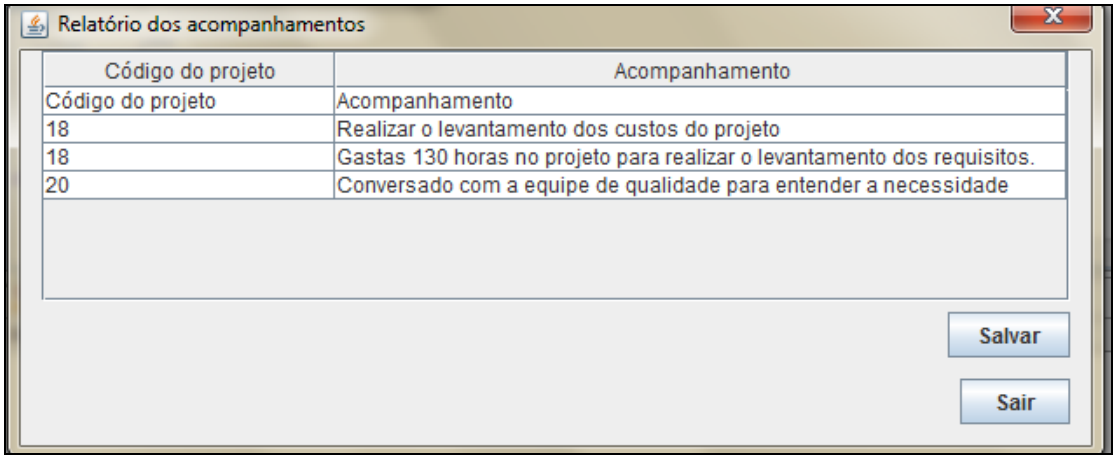


Figura 37 - Tela para gerar relatório de acompanhamento dos projetos

Para apresentar os dados na tela, é necessário realizar a busca dos registros no banco de dados. Essa busca é feita pelo método `buscaListaTodosAcompanhamentos` da classe `PesquisarAcompanhamentoDAO` conforme pode ser visto no Quadro 45.

```
public ArrayList<Acompanhamento> buscaListaTodosAcompanhamentos() throws
Exception {
       try {
              Connection connection;
              ConexaoBD conexao = ConexaoBD.getIntancia();
              connection = conexao.getConexao();
              Statement stm = connection.createStatement();
              resultPesquisa = stm.executeQuery("select * from
acompanhamento");

   while (resultPesquisa.next()) {
       Acompanhamento novoAcompanhamento = new Acompanhamento();
       novoAcompanhamento.setCodigoProjeto(Integer.parseInt(resultPesquisa.getS
tring("IDPROJETO")));
       novoAcompanhamento.setDsAcompanhamento(resultPesquisa.getString("DSACOMP
ANHAMENTO"));
       listaTodosAcompanhamentos.add(novoAcompanhamento);
   }

       } catch (SQLException ex) {
              return listaTodosAcompanhamentos;
       }
       return listaTodosAcompanhamentos;
}
```

Quadro 45 - Método para buscar todos os acompanhamentos

O método `buscaListaTodosAcompanhamentos` retorna uma lista com o código do projeto e a descrição dos acompanhamentos cadastrados. Essa lista é utilizada para preencher a tabela utilizada pela tela da Figura 37. Os dois métodos que são responsáveis pela montagem da tabela são `montaDados` e `adicionaTable`. O método `montaDados` é responsável por chamar o método `buscaListaTodosAcompanhamentos` para buscar as informações do banco de dados e chamar o método `adicionaTable` passando como parâmetro o código do projeto e a descrição do acompanhamento. O método `adicionaTable` por sua vez é responsável por adicionar uma

nova linha na tabela com as informações do código do projeto e a descrição do acompanhamento, conforme pode ser visto no Quadro 46.

```
public void montaDados(DefaultTableModel modelo) throws Exception{
	PesquisarAcompanhamentoDAO acompanhamento = new
PesquisarAcompanhamentoDAO();
	ArrayList<Acompanhamento> listaAcompanhamentos =
acompanhamento.buscaListaTodosAcompanhamentos();
	for(int i = 0; i < listaAcompanhamentos.size(); i++){
		String codProjeto =
String.valueOf(listaAcompanhamentos.get(i).getCodigoProjeto());
		String dsAcompanhamento =
listaAcompanhamentos.get(i).getDsAcompanhamento();
	adicionaTable(modelo, codProjeto, dsAcompanhamento);
	}
}

private void adicionaTable(DefaultTableModel modelo, String parmCodProjeto,
String parmDsAcompanhamento) throws SQLException {
	modelo.addRow(new Object[] {
			parmCodProjeto,
			parmDsAcompanhamento});
}
```

Quadro 46 - Montagem da tabela com os acompanhamentos

Após a montagem da tabela é possível salvar o relatório com os acompanhamentos dos projetos utilizando o método `arquivoPDF` explicado no Quadro 38.

3.3.2.6 Importar XML

Para realizar a importação do arquivo XML, é chamado o método `lerXML` da classe `Serializacao`. Esse método também utiliza a biblioteca *XStream* e através do método `fromXML` cria a classe Serializacao com as informações que foram exportadas (Quadro 47).

```
public static void lerXml(String local) {
	FileReader reader = null;
	try {
	// carrega o arquivo XML para um objeto reader
		reader = new FileReader(local);
	} catch (FileNotFoundException e) {
		e.printStackTrace();
	}
	// Cria o objeto xstream
	XStream xStream = new XStream(new DomDriver());
	instancia = (Serializacao) xStream.fromXML(reader);
}
```

Quadro 47 - Método para ler o arquivo XML

A classe `Serializacao` com as listas dos projetos, grupos de critérios, critérios selecionados e os pesos informados, é utilizada na criação dos *Table Model* para mostrar as

informações na tela de comparação dos projetos. Por exemplo, no método `getValueAt` da classe `GrupoTableModel` é verificado se existe uma instância da classe `Serializacao` criada. Se existe, significa que foi importado um arquivo XML e por isto devem ser utilizadas as informações contidas no arquivo. Para saber se o grupo de critério deve ou não ser selecionado, é verificado se o código do grupo de critério existe no *HashMap* da `Serializacao` (Quadro 48). Esse mesmo procedimento é realizado no *Table Model* do projeto e critério.

```
public Object getValueAt(int rowIndex, int columnIndex) {
   if (rowIndex > -1 && rowIndex < listaGrupos.size()) {
       GrupoCriterio grupo = listaGrupos.get(rowIndex);
        switch (columnIndex) {
          case 0:
             Serializacao serializacao = Serializacao.getIntancia();
                  if(serializacao == null){
                         return grupo.getEstaSelecionado();
                  }else{

                      if(serializacao.getListaGruposCriteriosSelecionados().contains
                      Key(grupo.getCodigoGrupo())){
                             listaGruposSelecionados.add(grupo);
                             return true;
                      }else{
                             return false;
                      }
                  }
           case 1:
             return grupo.getNomeGrupo();
           default:
             return null;
           }
   }
}
```

Quadro 48 - Seleciona o grupo de critério na comparação de projetos

Já para os pesos informados, a verificação de se existe uma instância da classe `Serializacao` é feita quando são atribuídos os valores para os vetores que guardam os pesos informados (Quadro 49). Esse mesmo procedimento é feito para o *Table Model* dos pesos entre os grupos de critérios, os pesos entre os critérios e o peso entre os projetos para os grupos de critério e critérios.

```
if(serializacao == null){
       if(i == j){
              matrizValores[i][j] = 1;
       }else{
              matrizValores[i][j] = 0;
       }
}else{
       String nomeGrupo = listaGruposSelecionados.get(i+1).getNome()
       + listaGruposSelecionados.get(j+1).getNome();
       if(serializacao.getListaPesoGrupos().containsKey(nomeGrupo)){
        matrizValores[i][j] = serializacao.getListaPesoGrupos().get(nomeGrupo);
       }
}
```

Quadro 49 - Informar peso conforme arquivo XML

3.4 RESULTADOS E DISCUSSÃO

Em relação aos trabalhos correlatos apresentados na fundamentação teórica, o Quadro 50 apresenta o comparativo entre a aplicação desenvolvida e as duas ferramentas descritas nos trabalhos correlatos. Os outros dois trabalhos apresentados nos trabalhos correlatos, Metodologia de apoio à decisão para priorização de projeto (PINHO, 2006) e Priorização de projetos, através de identificação e análise de critérios de seleção (CASTRO, 2010) foram utilizados para compreender o cálculo realizado pela técnica AHP e a sua execução.

Aplicação / Característica	IPÊ 1.0 (Costa, 2004)	My choise, my decision (Init, 2007	Priorização e seleção de projeto
Salva as informações cadastradas	Sim	Sim	Sim
Banco de dados	Não utiliza	Não utiliza	Oracle
Cadastro de grupos de critério e critérios	Sim	Não	Sim
Acompanhamento de projetos	Não	Não	Parcial
Idioma	Português / Inglês	Inglês	Português
Plataforma da ferramenta	Desktop	Web	Desktop
Aplicação da técnica AHP	Sim	Sim	Sim

Quadro 50 - Comparativo entre as características da aplicação e trabalhos correlatos

A aplicação desenvolvida alcançou os objetivos propostos e agregou novas funcionalidades em relação às ferramentas correlatas, permitindo tanto a aplicação da técnica AHP quanto o acompanhamento de projetos.

Um dos diferenciais da aplicação desenvolvida é permitir o acompanhamento dos projetos, mesmo que de forma parcial e limitada. Enquanto as aplicações *My choise, my decision* desenvolvido pela empresa Init (2007), e a aplicação *IPÊ 1.0* de Costa (2004) não possuem o acompanhamento dos projetos, com a aplicação desenvolvida é possível realizar o cadastramento das atividades executadas nos projetos e realizar o seu acompanhamento. Portanto, a aplicação proposta além de realizar a seleção e priorização de projetos, apóia a gestão de portfólio, com o acompanhamento dos projetos e a geração de relatórios com a priorização dos projetos, com o peso dos critérios informados e as informações cadastradas nos projetos.

Já em relação a salvar as informações, é possível afirmar que todas as aplicações analisadas atendem a esse requisito. Na aplicação de Costa (2004) as informações são salvas em um arquivo localizado na máquina onde a aplicação está rodando. Já a aplicação desenvolvida pela empresa Init (2007) salva as informações na conta do usuário conectado.

Por fim, na aplicação proposta neste trabalho, as informações são salvas em banco de dados Oracle, enquanto as outras aplicações analisadas não utilizam banco de dados.

Levando tem conta os trabalhos correlatos, percebe-se que um fator que pode sofrer melhorias em futuros trabalhos é a usabilidade da aplicação. Enquanto na aplicação da empresa Init (2007) informa os pesos durante o processo de comparação é apenas necessário selecionar o botão com o peso, na aplicação desenvolvida neste trabalho é necessário selecionar o campo e digitar manualmente o valor. Entretanto, deve ficar claro também que em função da forma como o sistema da empresa Init foi desenvolvido isso acaba tornando-se uma limitação da ferramenta, uma vez que será possível apenas informar valores entre as faixas mostradas na aplicação.

Durante a concepção deste trabalho e elaboração de sua proposta, pensou-se se os valores informados durante a comparação dos projetos deveriam ser salvos. Optou-se por salvar as informações em arquivos XML e possibilitar a importação desses arquivos após a finalização da comparação. Esse recurso torna-se um facilitador da aplicação uma vez que caso os pesos informados durante a comparação mudem, não é necessário realizar todo o cadastro novamente.

Também durante a concepção da proposta deste trabalho, pensou-se em realizar a criação de uma aplicação que além de realizar a seleção e priorização de projetos, possa auxiliar os gerentes de projetos no acompanhamento das atividades dos projetos. Para isso foi criado um recurso que possibilita que o usuário cadastre as atividades nos projetos, podendo posteriormente gerar relatórios com todas as atividades cadastradas nos projetos.

Uma limitação do trabalho desenvolvido consiste em possuir apenas integração com o banco de dados Oracle. Para trabalhos futuros pode-se realizar a integração com mais bancos de dados, principalmente com o SQL Server, que é um banco de dados muito utilizado pelas organizações. Sugere-se também a utilização de arquivos textos, ou XML, o que acaba não obrigando a instalação de um banco de dados para a utilização da aplicação.

A principal dificuldade encontrada durante o desenvolvimento deu-se com relação à automatização da aplicação da técnica AHP, uma vez que exigiu um alto conhecimento na técnica e a realização de vários cálculos subseqüentes, o que acabou necessitando um grande volume de armazenamento de cálculos.

Conclui-se por fim, que com a elaboração deste trabalho, torna-se possível a organização decidir qual projeto deve ser priorizado e acompanhar as atividades de cada projeto, o que acaba auxiliando na supervisão dos projetos durante todas as suas fases de construção.

4 CONCLUSÕES

Após o estudo sobre o cenário atual do ambiente corporativo, percebe-se que a globalização e a concorrência proveniente das relações comerciais acarretam mudanças rápidas, demandando o desenvolvimento e a implantação de projetos cada vez mais modernos e que envolvem atividades com alto nível de complexidade por parte das organizações.

Neste contexto, a gestão de portfólio tem sido considerada uma boa estratégia para empresas que necessitam saber quais projetos devem ser priorizados, uma vez que seus recursos são limitados e por conta disso, a demanda de projetos é superior a sua capacitação de realização.

Em problemas de decisões complexas, geralmente vários critérios podem ser necessários para uma escolha final entre diferentes alternativas. Os métodos de apoio multicritério à decisão têm o objetivo de suportar, conduzir e esclarecer o processo decisório, através de uma modelagem matemática. Dentre estes métodos, o método AHP em específico destaca-se por ser o mais utilizado. No entanto, este método possui uma complexidade alta e desta forma exige um grau de conhecimento elevado e demanda muito esforço para sua aplicação quando esta é realizada manualmente ou sem a ajuda de ferramentas adequadas.

Portanto, o desenvolvimento desta aplicação auxilia na automatização do processo de seleção e priorização de projetos, disponibilizando informações para que a tomada de decisão ocorra de maneira mais segura e precisa, sem a necessidade de realizar vários cálculos manuais ou possuir várias planilhas para realizar a comparação entre os projetos.

A utilização do Oracle XE como banco de dados foi satisfatória, uma vez que a aplicação não possui um alto volume de dados e por isto não precisa de uma performance excelente.

A decisão de desenvolver uma aplicação desktop teve como objetivo atender à grande maioria das organizações existentes, já que nos dias atuais as empresas possuem uma área específica para realizar o gerenciamento de projetos com acesso a computador.

Em relação aos objetivos propostos inicialmente, foi disponibilizada uma aplicação que implementa o algoritmo AHP para seleção e priorização dos projetos e apóia as atividades de seleção e acompanhamento de projeto. O projeto abrange qualquer área, não sendo restrito apenas a área de TI, como havia sido proposto nos objetivos. Assim sendo, o estudo dos conceitos e técnicas aplicadas ao trabalho proporcionou o desenvolvimento, atendendo com sucesso os objetivos inicialmente estabelecidos.

4.1 EXTENSÕES

Algumas sugestões de extensão para este trabalho são:

a) disponibilizar a aplicação também para o ambiente web, para aumentar a portabilidade do sistema;
b) melhoria e aperfeiçoamento no mecanismo utilizado para informar os pesos durante a comparação de projetos, com o intuito de melhorar a usabilidade da aplicação;
c) realizar a integração da aplicação também com o banco de dados SQL Server, para aumentar a portabilidade do sistema;
d) possibilitar um melhor acompanhamento de projeto, sendo possível cadastrar informações como escopo, custo, tempo e qualidade;
e) possibilitar a avaliação dos projetos observando se ainda são viáveis e aderentes aos critérios pelos quais foram aprovados.

REFERÊNCIAS BIBLIOGRÁFICAS

CASTRO, Eduardo M. **Priorização de projetos, através da identificação e análise de critérios de seleção, selecionados aos objetivos estratégicos de negócio**. 2010. 88 f. Dissertação (Mestrado em Executivo em Gestão Empresarial) – Escola Brasileira de Administração Pública de Empresas, Fundação Getúlio Vargas, Rio de Janeiro.

COSTA, Helder G. **IPÊ:** guia do usuário. 2004. 26 f. Relatórios de Pesquisa em Engenharia de Produção da UFF (Mestrado em Engenharia de Produção) – Departamento de Engenharia de Produção, Universidade Federal Fluminense, Niterói.

INIT Grupa. [S.l.], [2007?]. Disponível em: <http://www. http://123ahp.com/l>. Acesso em: 10 nov. 2012.

KOONTZ, Harold; O'DONNEL, Cyril. **Os princípios de administração:** uma análise das funções administrativas. São Paulo: Pioneira, 1980.

LEVINE, Harvey. **Project portfolio management –** a practical guide to selecting projects, managing portfolios, and maximizing benefits. San Francisco: John Wiley & Sons, 2005.

MAIZLISH, Bryan; HANDLER, Robert. **IT portfolio management step-by-step.** Hoboken: John Willey & Sons, 2005.

MODICA, Jose E.; ROQUE JUNIOR, Rabechini; BRAUN, Edison M. Priorização de um portfólio de projetos. In: SEMINÁRIO INTERNACIONAL DE GERENCIMENTO DE PROJETOS, 11., 2011, São Paulo. **Anais eletrônicos...** São Paulo: PMI, 2011. p. 1-10. Disponível em: <http://pmisp.pmisp.org.br/sites/default/files/works/priorizacao_de_um_portfolio_de_projetos.pdf>. Acesso em: 18 abr. 2012.

PINHO, Selma F. C. **Uma metodologia de apoio à decisão para priorização de projetos de tecnologia da informação**. 2006. 163 f. Tese (Doutorado em Engenharia Civil) – Programa de Pós-Graduação de Engenharia, Universidade Federal do Rio de Janeiro, Rio de Janeiro.

PROJECT MANAGEMENT Institute. [S.l.], [2008]. Disponível em: <http://www.pmi.org.br/portal/sobre-o-pmi.html>. Acesso em: 31 mar. 2012.

RUSSO, Eduardo; SCHOEMAKER, Paul J. H. **Tomada de decisões:** armadilhas. São Paulo: Saraiva, 1993.

SAATY, Thomas L. **Theory and applications of the analytic network process:** decision making with benefits. Pittsburgh: RWS Publications, 2005.

SARDAGNA, Marcelo; VAHLDICK, Adilson. **Aplicação do Padrão Data Access Object (DAO) em Projetos Desenvolvidos com Delphi**. Blumenau, 2008. Disponível em: < http://www.inf.ufsc.br/erbd2008/artigos/2.pdf >. Acesso em: 12 dez. 2012.

SILVA, Amanda C. S.; NASCIMENTO, Leila P. A.; BELDERRAIN, Mischel C. N. Método de apoio multicritério à decisão na seleção e priorização de portfólio de projetos. In: ENCONTRO DE INICIAÇÃO CIENTÍFICA E PÓS-GRADUAÇÃO DO ITA, 13., 2007, São José dos Campos. **Anais eletrônicos...** São José dos Campos: ITA, 2007. p. 1-9. Disponível em: <http://www.bibl.ita.br/xiiiencita/MEC13.pdf>. Acesso em: 18 abr. 2012.

SOMMERVILLE, Ian. **Engenharia de software.** 6. ed. São Paulo: Addison Wesley, 2003.

VARGAS, Ricardo A. Utilizando a programação multicritérios (Analytic Hierarchy Process - AHP) para selecionar e priorizar projetos na gestão de portfólio. In: PMI GLOBAL CONGRESS, 11., 2010, Washington. **Anais eletrônicos...** Washington: PMI, 2010. p. 1-22. Disponível em: <http://issuu.com/ricardo.vargas/docs/ahp_project_selection_pt>. Acesso em: 18 abr. 2012.