

**UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
DEPARTAMENTO DE INFORMÁTICA E ESTATÍSTICA
CURSO DE SISTEMAS DE INFORMAÇÃO**

**Portabilização da Ferramenta de Modelagem de Banco de Dados Relacional
brModelo**

Alunos: Leonardo Antonio Ramos e Otávio Soares Menna
Orientador: Ronaldo Santos Mello

Florianópolis - SC
2011/1

Sumário

Sumário.....	2
Lista de Figuras.....	3
Lista de Siglas.....	5
1. Introdução	6
2. Projeto de Bancos de Dados Relacionais.....	8
2.1 Projeto Conceitual	8
2.2 Projeto Lógico	10
2.3 Projeto Físico	11
3. Ferramenta brModelo	13
4. brModeloNext	16
4.1 Funcionalidades	17
4.2 Análise e Planejamento da Interface com o Usuário da brModelo	26
5. Conclusão	30
6. Referências Bibliográficas	31
7. Apêndice	31
7.1 Código-fonte.....	10
7.2 Artigo.....	11

Lista de Figuras

Figura 1: Exemplo de esquema ER.....	9
Figura 2: Tela da brModelo apresentando um exemplo de modelagem conceitual.....	13
Figura 3: Tela da brModelo apresentando um exemplo de modelagem lógica.	14
Figura 4: Semelhança entre a representação de grafos (acima) e a representação de modelagem conceitual de dados (abaixo)	17
Figura 5: Tela principal da brModeloNext	18
Figura 6: Tela para uma modelagem lógica na brModeloNext.....	19
Figura 7: Informações sobre o objeto através de tooltip e barra de status	20
Figura 8: Descrição destacada do passo a ser executado para a inserção do objeto no modelo através da barra de status	20
Figura 9: Exemplo de uma tentativa de incluir um objeto de forma equivocada na modelagem e o aviso alertando o erro cometido pelo usuário	21
Figura 10: Visualização de 2 modelagens na mesma tela	22
Figura 11: Menu “Modelagem conceitual”.....	23
Figura 12: Menu “Modelagem lógica”	23
Figura 13: Janela de edição in-place, acionada ao inserir o objeto na modelagem e também ao realizar um duplo-clique sobre o objeto	24
Figura 14: Descrição detalhada do objeto exibida em uma tooltip ao fixar o ponteiro do mouse no objeto.....	24
Figura 15: Exemplo de interatividade entre a brModeloNext e o usuário no processo de mapeamento de um relacionamento e seus atributos.....	25
Figura 16: Exemplo de interatividade entre a brModeloNext e o usuário no processo de mapeamento de uma especialização	25
Figura 17: Modelagem das classes de controle da BrModeloNext.....	25
Figura 18: Modelagem das classes de entidade da BrModeloNext.....	25
Figura 19: Modelagem das classes de entidade da BrModeloNext.....	25

Lista de Siglas

BD - Banco de dados

SGBD - Sistema de Gerência de BD

GBD/UFSC - Grupo de BD da UFSC

XML - eXtensible Markup Language

SQL - Structured Query Language

ER – Modelo Entidade-Relacionamento

1. Introdução

Vivemos em uma época na qual dependemos da informação para tudo o que fazemos. Produzimos e consumimos uma grande carga de informação ao longo da nossa vida. Uma maneira de armazenar e gerenciar informação é através de bancos de dados. Bancos de dados (BDs) são essenciais para a organização, armazenamento e tratamento de dados [1].

Projetar BDs é uma tarefa extremamente importante para garantir a sua qualidade [2]. Construir um BD sem considerar uma metodologia de projeto, principalmente quando se trata de um domínio de aplicação complexo, faz com que o banco corra o risco de apresentar problemas como falta de consistência de dados, armazenamento de dados com redundância e difícil manutenibilidade do banco, dentre outros. É primordial para o sucesso de um BD que ele passe por uma etapa de abstração do domínio de aplicação (modelagem conceitual dos dados), identificando todas as entidades envolvidas no domínio e os relacionamentos existentes entre elas, e uma etapa de transformação desta abstração para um esquema lógico de tabelas (modelagem lógica). A execução destas etapas evita ou reduz os problemas mencionados anteriormente, além de facilitar a validação do desenvolvimento do projeto junto aos usuários.

Existem diversas ferramentas que auxiliam no projeto de BDs relacionais, porém, as ferramentas comercialmente disponíveis tratam a modelagem de forma atrelada ao Sistema de Gerência de BD (SGBD), ou seja, apenas os níveis lógico e de implementação em SQL (nível físico) são considerados [3, 4, 5]. A inexistência de ferramentas completas de projeto de BD, que consideram os três níveis de modelagem – conceitual, lógico e físico - e que tenham propósitos educacionais, além do uso comercial, motivou o Grupo de BD da UFSC (GBD/UFSC) [6] a desenvolver a ferramenta *brModelo* [7].

Além do diferencial recém mencionado, ou seja, da consideração de todos os níveis (ou etapas) de modelagem (conceitual, lógico e de implementação) no projeto *top-down* de um BD relacional, a *brModelo* propõe uma solução amigável e didática. Este objetivo é alcançado através de um processo o mais automático possível de conversão entre etapas de modelagem, como também do questionamento ao usuário quando há mais de uma possibilidade de conversão conceitual-lógico para um determinado conceito. Esta característica não é encontrada em ferramentas comerciais, que geralmente realizam um processo totalmente automático de conversão, gerando um BD nem sempre adequado às intenções do projetista. Esta interação com o usuário permite que o mesmo se familiarize com os detalhes do processo de projeto de um BD, adquirindo experiência [2].

Desde a liberação da sua primeira versão, em meados de 2005, a brModelo tem tido uma ótima procura por parte do meio acadêmico e outros interessados, principalmente para aprendizado de modelagem de BD. Entretanto, apesar da sua popularidade, a brModelo só atende plataformas que executam o sistema operacional Windows e apresenta ainda alguns erros de mapeamento entre etapas, bem como algumas limitações. Na etapa de modelagem conceitual, por exemplo, a ferramenta não permite que uma entidade participe de mais de um auto-relacionamento e não permite combinar hierarquias de especialização distintas (exclusivas e não-exclusivas) para uma mesma entidade. Já na etapa de modelagem lógica, ela apresenta problemas algumas vezes na geração e posicionamento de chaves estrangeiras e atributos de relacionamento para alguns tipos de cardinalidade de relacionamento.

Assim sendo, este trabalho tem por objetivo reimplementar a brModelo visando sanar os erros existentes na versão atual, além de revisar e melhorar a sua usabilidade com o auxílio de experimentos com usuários e também deixá-la independente de plataforma através do uso da tecnologia Java [8].

Um maior detalhamento sobre este trabalho é exposto nos capítulos seguintes. O capítulo 2 apresenta a fundamentação teórica que envolve este trabalho: projeto de BD. O capítulo 3 descreve a ferramenta que é o objeto de estudo deste trabalho, apontando suas características, bem como vantagens e desvantagens da versão atual. O capítulo 4 trata sobre a ferramenta em desenvolvimento, as tecnologias envolvidas na sua construção e as suas funcionalidades. O capítulo 5 aborda as considerações finais sobre o desenvolvimento deste trabalho.

2. Projeto de Bancos de Dados Relacionais

A brModelo é uma ferramenta para projeto de BD relacional. Assim sendo, ela abrange todas as etapas, tarefas e subtarefas necessárias neste processo.

Um projeto de BD passa por três níveis de modelagem, sendo eles [2]:

- 1- Modelagem conceitual (ou projeto conceitual), onde é definida uma abstração de dados em alto nível. O objetivo desta etapa é representar os requisitos de dados de um domínio através da descrição dos fatos relevantes do mundo real e dos seus relacionamentos. Este alto nível de abstração permite a construção de uma modelagem de dados independente do modelo de BD a ser utilizado posteriormente pela aplicação;
- 2- Modelagem lógica (ou projeto lógico), onde a representação da modelagem conceitual é mapeada para uma modelagem relacional de BD. Nesta etapa enfatiza-se a eficiência no armazenamento, ou seja, evita-se a criação excessiva de tabelas, sub-utilização de tabelas, etc.;
- 3- Modelagem física (ou implementação), onde um esquema SQL correspondente é gerado para a modelagem lógica. Esta etapa é completamente dependente do SGBD escolhido e seu dialeto SQL específico. Nesta fase, o desenvolvedor deve dar maior ênfase à eficiência de acesso, ou seja, na implementação de consultas, *views* e índices que possibilitem maior performance no acesso a dados¹.

A migração entre uma etapa e outra pode ser feita de forma automatizada, porém, um bom processo de modelagem é aquele que permite ao projetista de BD decidir por uma dentre diversas alternativas de mapeamento, quando possível. As próximas Seções detalham cada uma dessas três etapas de desenvolvimento de um projeto de BD.

2.1 Projeto Conceitual

Um projeto conceitual de BD tem as seguintes premissas:

- 1- deve ser independente de detalhes de implementação de qualquer SGBD. Desta forma, há maior facilidade na compreensão da semântica dos dados de um domínio, inclusive para usuários leigos, que estarão muito envolvidos no processo até este momento;
- 2- deve permitir seu mapeamento para qualquer modelo lógico de BD, como por exemplo, BD relacional, BD orientado a objetos ou BD XML;
- 3- deve facilitar a manutenção do modelo lógico, bem como sua migração para algum outro modelo lógico. Neste sentido, é preferível realizar estas tarefas sobre um

¹ Vale salientar que a brModelo não oferece suporte para todas estas implementações. Atualmente, ela apenas gera o esquema de implementação do BD através de um conjunto de comandos CREATE TABLE da SQL.

modelo mais abstrato, pois a semântica é mais clara e a modificação de requisitos é facilitada.

O modelo mais utilizado para modelagem conceitual de BD é o modelo Entidade-Relacionamento (ER) [2]. A vantagem da utilização deste modelo reside na sua simplicidade. Ele possui conceitos simples, representação gráfica reduzida e de fácil compressão. Um projeto conceitual realizado neste modelo é chamado de Esquema ER ou Diagrama ER.

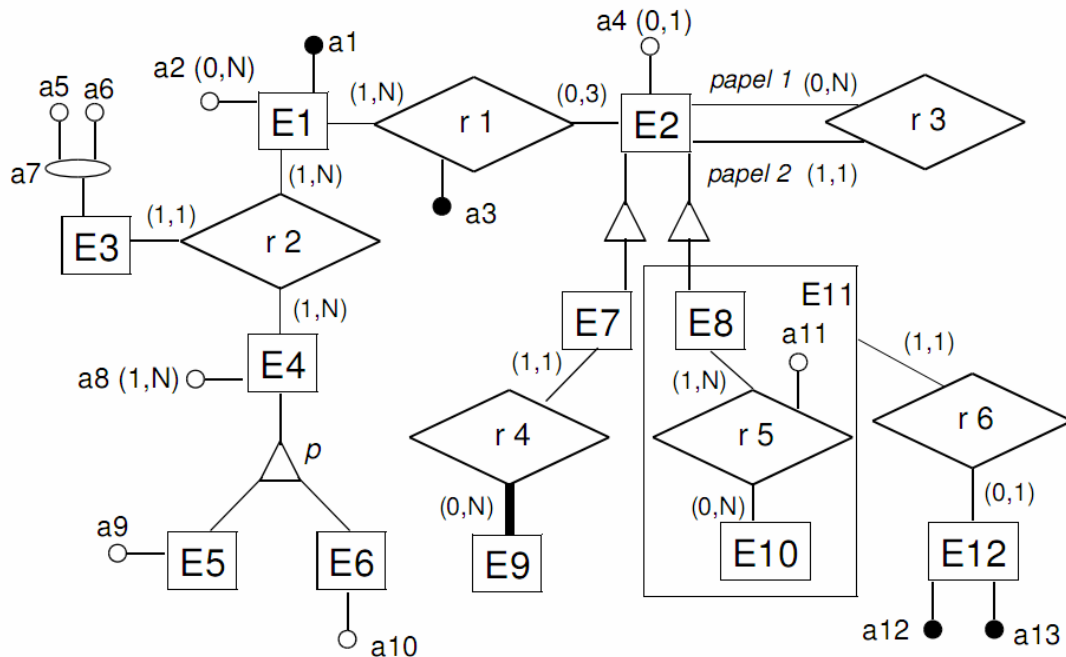


Figura 1: Exemplo de esquema ER.

A Figura 1 mostra um exemplo de esquema ER na notação utilizada pela brModelo. Com base nesta Figura, os principais conceitos do modelo, descritos a seguir, são exemplificados:

Uma *Entidade* é uma abstração de um fato no mundo real para o qual se deseja manter seus dados no BD. Ela é simbolizada no diagrama ER através de um retângulo nomeado. *E1* e *E10* são exemplos.

Um *Relacionamento* é uma abstração de uma associação entre ocorrências de entidades, sendo simbolizado por um losango nomeado. A definição de um relacionamento envolve a definição da cardinalidade máxima (a quantidade máxima de ocorrência de entidades que podem estar associadas a uma ocorrência de outra entidade) e da cardinalidade mínima, que indica se a participação das ocorrências de entidade no relacionamento é obrigatória ou opcional. Um relacionamento clássico entre 2 entidades é conhecido como relacionamento binário, como por exemplo, r1 e r5. Tem-se ainda o auto-relacionamento (representado por r3), que representa uma associação entre ocorrências de uma mesma entidade, exigindo a identificação de papéis ou estereótipos, e o relacionamento n-ário, uma abstração de uma associação entre "n"

ocorrências de entidades. O exemplo mais típico é o relacionamento ternário, como por exemplo, o relacionamento r2.

Atributo é a abstração de uma propriedade de uma entidade ou de um relacionamento. Atributos são classificados em obrigatórios / opcionais (a8 / a4), monovalorados / multivalorados (a4 / a2) e simples / compostos (a9 / a7). Esta classificação altera sua representação simbólica no modelo ER. Atributos podem ser identificadores, ou seja, distinguem ocorrências de uma entidade umas das outras, garantindo o acesso individualizado a uma ocorrência de entidade no BDs. Neste contexto, entidades fracas são entidades cuja identificação de suas ocorrências depende da identificação de outras entidades. Um exemplo de entidade fraca é *E9*. Ela é indicada como fraca devido ao relacionamento com um traço mais grosso que a conecta com a sua entidade forte (*E7*). Neste caso, a identificação de *E7* é necessária para definir a identificação de *E9*.

A *especialização* ocorre quando entidades definem atributos e/ou relacionamentos particulares a um subconjunto de ocorrências de uma entidade genérica, com herança de propriedades entre uma entidade e outra. Especializações podem ser totais / parciais e exclusivas / não-exclusivas (onde uma ocorrência de entidade genérica pode ter mais de uma especialização). De *E4* para *E5* e *E6*, tem-se uma especialização parcial e exclusiva. De *E2* para *E8*, uma especialização total e de *E2* para *E8* e *E7*, uma especialização não-exclusiva.

Uma entidade associativa abstrai uma associação entre entidades, necessária quando ocorrências de um relacionamento devem se relacionar com outras entidades. Sua representação é feita através de um retângulo circundando o losango do relacionamento ou então um retângulo envolvendo o relacionamento e as entidades conectadas a ele, como é o caso da entidade *E11* na Figura 1

2.2 Projeto Lógico

Ao contrário do modelo conceitual, o modelo lógico já leva em consideração os limites impostos por algum tipo de tecnologia de BD, seja ele relacional, orientado a objeto ou qualquer outro. Esta etapa é responsável pelo mapeamento de um esquema conceitual para o esquema lógico alvo. Várias formas de estruturar uma modelagem conceitual, abstrata, em um esquema relacional são possíveis.

Nesta etapa existe um compromisso entre evitar:

- um grande número de tabelas. O objetivo aqui é evitar um longo tempo de resposta em consultas e atualizações de dados, ou seja, evitar junções excessivas;
- muitos atributos opcionais, para evitar desperdício de espaço em disco;
- muitos controles de integridades no BDs, o que implica em evitar muitas dependências entre dados.

O mapeamento de uma modelagem conceitual para uma modelagem lógica pode ser realizado em três etapas:

1- Mapeamento preliminar de entidades e seus atributos: etapa na qual as entidades do esquema conceitual são convertidas em tabelas e seus atributos simples são convertidos em colunas de tabelas. No caso de entidades fracas, o atributo identificador da entidade forte torna-se parte da chave primária na tabela correspondente à entidade fraca, bem como uma chave estrangeira para a tabela que gerou a entidade forte.

2- Mapeamento de especializações: nesta etapa, três alternativas são geralmente adotadas: uma tabela única para a entidade genérica e suas especializações, uma tabela para a entidade genérica e uma tabela para cada entidade especializada, ou ainda, criar tabelas apenas para as entidades especializadas. As regras de mapeamento de entidades utilizadas na etapa anterior são novamente aplicadas aqui.

3- Mapeamento de relacionamentos e seus atributos: nesta etapa, dependendo da cardinalidade atribuída ao relacionamento, algumas opções de conversão podem ser adotadas:

- no caso de relacionamentos com cardinalidade 1-1 (um para um), quando as entidades obrigatoriamente se relacionam em ambos sentidos, funde-se as entidades relacionadas em uma só tabela. Caso uma das entidades seja opcional, a mesma opção de uma só tabela é uma das alternativas, e a outra, é adicionar como uma chave estrangeira representando a entidade obrigatória na tabela da entidade opcional. Para a situação na qual ambas as entidades são opcionais, a primeira alternativa é gerar três tabelas, uma para o relacionamento e duas para as entidades e a segunda é adicionar os atributos do relacionamento em uma das duas tabelas de cada entidade.
- para relacionamentos com cardinalidade 1-n (um para n), se a entidade é obrigatória ou opcional do lado n e obrigatória do lado 1, deve-se criar chave estrangeira na tabela do lado n para representar o relacionamento. Já se a entidade é obrigatória ou opcional do lado n e opcional do lado 1, vale esta mesma opção anterior da chave estrangeira, ou ainda, gera-se uma tabela para representar o relacionamento.
- em relacionamentos com cardinalidade n-m (n para m) uma tabela para o relacionamento com chaves estrangeiras representando as entidades é criada.

2.3 Projeto Físico

O Projeto Físico de um BDs tem como principal objetivo otimizar o acesso a dados, ou seja, definir estratégias de acesso, como por exemplo, uma análise das consultas mais frequentes e consequente definição de *views* que processem estas consultas de maneira otimizada. As estratégias de acesso a dados devem ser frequentemente revistas durante o tempo de vida de um BD, já que o tamanho ou

mesmo estrutura do BD pode ser alterada, além de determinados tipos de acesso terem frequência variante de forma sazonal.

Vale salientar que nesta etapa ocorre a definição de índices. Índices são apontadores físicos no BDs, permitindo a localização mais rápida dos registros nas tabelas. Como os testes de atributos são feitos em memória durante o processamento das consultas, o acesso a dados é sensivelmente mais performático.

Recomenda-se o uso de índices para atributos que estão frequentemente envolvidos em predicados de junção ou seleção, bem como tabelas com grandes volumes de dados e alta frequência de acesso em consultas. Além disso, também indica-se o uso de índices em atributos que são chaves alternativas e envolvidos em transações críticas, que exigem um tempo de resposta pequeno. O mapeamento da modelagem lógica para física dá-se quase que diretamente, ou seja, as tabelas definidas no projeto lógico são especificadas no dialeto SQL do SGBD alvo. Porém, como o foco do projeto físico é a melhoria da performance, o processo realiza ainda e de anomalias de atualização.

Para melhoria de performance no mapeamento, pode-se, por exemplo:

- melhorar a definição de dados no sentido de evitar chaves compostas por diversos atributos, reduzindo o acesso a disco;
- implementar as consultas mais frequentes através de expressões SQL otimizadas;
- definir índices para os acessos mais frequentes e críticos.

3. Ferramenta brModelo

brModelo [7] é uma ferramenta de apoio ao projeto de um BD relacional desenvolvida pelo GBD/UFSC. Ela consiste em um aplicativo que permite a definição de modelagens conceituais e lógicas para BDs relacionais com facilidade e independência de SGBD. A principal finalidade do desenvolvimento da brModelo foi a de oferecer uma ferramenta voltada para o aprendizado de modelagem de BDs relacionais.

As principais propostas da brModelo são:

- basear-se na metodologia completa de projeto de BDs relacionais [2].
- ter como foco de implementação a modelagem conceitual, por se tratar de uma importante etapa na modelagem de BDs e visto que as ferramentas semelhantes não ofereceram essa alternativa.
- realizar o mapeamento da modelagem conceitual para a modelagem lógica, bem como o mapeamento da modelagem lógica para a modelagem física.
- criação de dicionários de dados completos

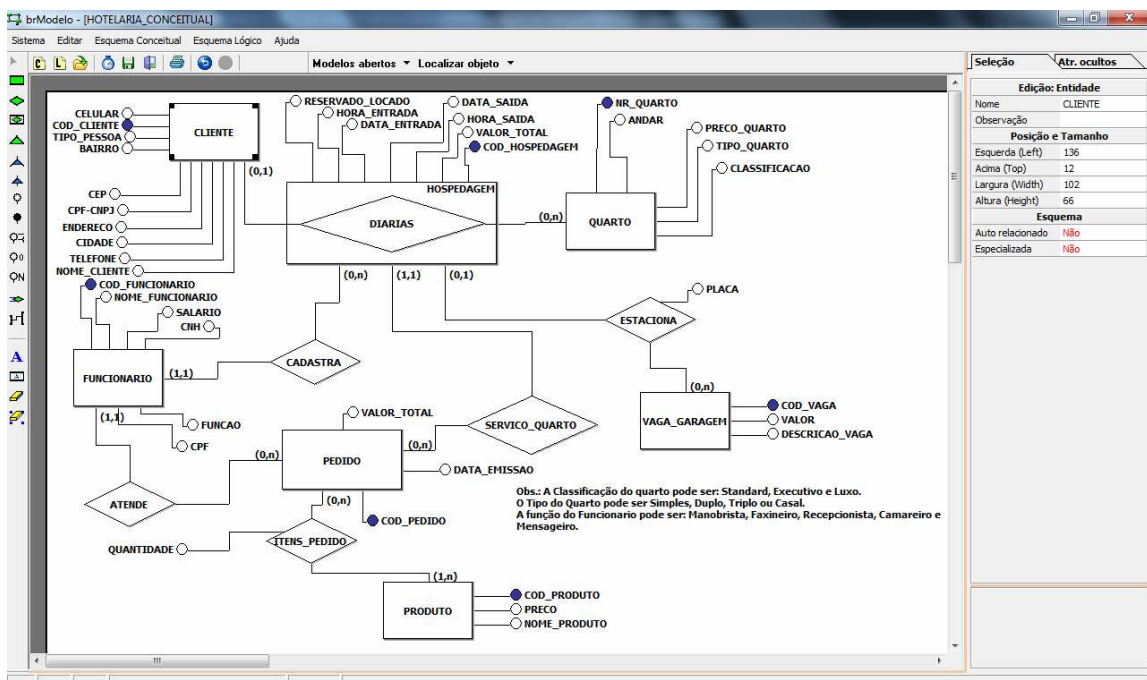


Figura 2: Tela da brModelo apresentando um exemplo de modelagem conceitual.

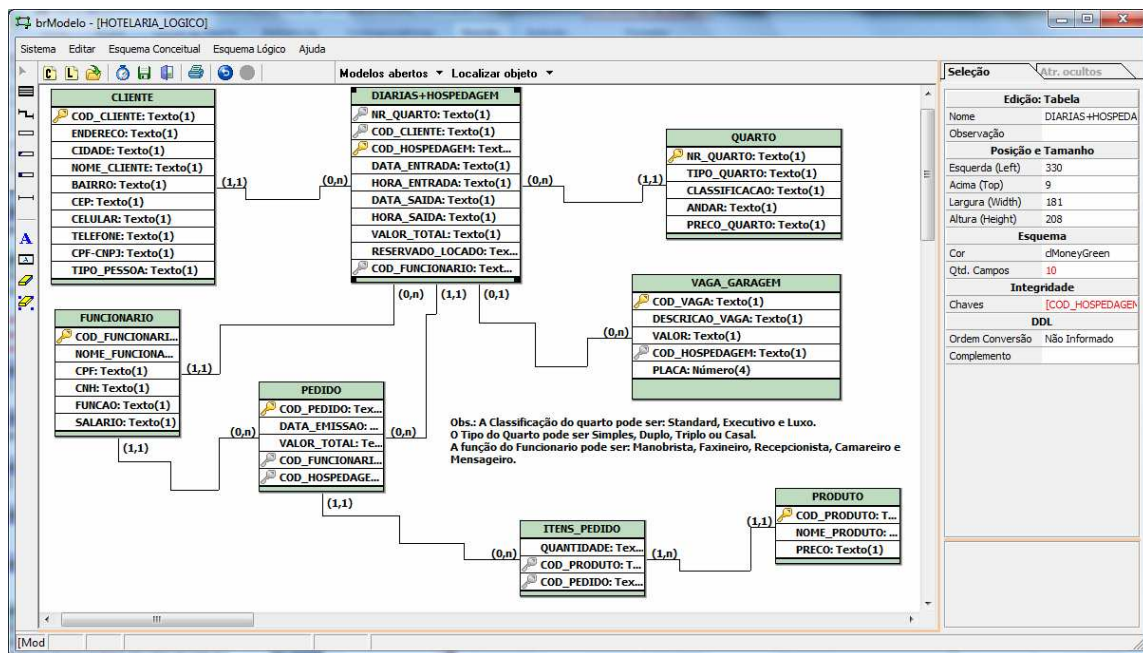


Figura 3: Tela da brModelo apresentando um exemplo de modelagem lógica.

As Figuras 2 e 3 mostram exemplos de uma modelagem conceitual e de uma modelagem lógica realizadas na brModelo. A interface da brModelo é composta basicamente por uma representação de página (semelhante a editores de texto) utilizada para a criação do desenho da modelagem. Uma barra vertical à esquerda contém todos os elementos envolvidos no modelo em edição no momento (conceitual ou lógico). Estes elementos podem ser selecionados e inseridos no desenho da modelagem utilizada. Na parte direita da tela existe um inspetor de elementos, que exibe as características de um ou mais elementos selecionados na modelagem e possibilita a edição destas características.

A brModelo possui ainda uma barra de ferramentas e uma barra de menu, através das quais é possível acessar funções comuns a muitos sistemas, como funções de gerenciamento de arquivos e gerenciamento de conteúdo, bem como outras funções relativas ao processo de modelagem de BD, como por exemplo, o mapeamento entre os modelos.

A primeira versão da brModelo foi liberada em 2005 e desde então ela tem sido bastante procurada para fins de aprendizagem em modelagens para BD, principalmente por instituições de ensino brasileiras. Dentre os pontos fortes estão a independência de SGBDs e a consideração da modelagem conceitual no projeto. O fato da ferramenta ser fortemente baseada nos conceitos do modelo ER e facilita o processo de modelagem, pois o modelo ER dispõe de uma representação gráfica de fácil compreensão e com conceitos simples.

A brModelo foi implementada através da linguagem Object Pascal / Delphi, tornando sua execução possível somente em sistemas operacionais Windows. Visto que

se trata de uma ferramenta com propósitos acadêmicos, este é um fator caracterizado como a principal desvantagem. Algumas outras desvantagens da brModelo:

- visualização da modelagem nem sempre é limpa, ou seja, ela apresenta, algumas vezes, problemas de desenhos sobrepostos que dificultam a legibilidade;
- edição das características dos elementos presentes no desenho das modelagens só pode ser realizada através do inspetor de elementos
- alguns erros de consistência na conversão entre modelos, conforme salientado no Capítulo 1;
- algumas limitações na definição de esquemas conceituais, conforme salientado também no Capítulo 1.

Estas limitações motivaram o desenvolvimento de uma nova versão desta ferramenta, sendo este o foco deste trabalho de conclusão de curso. Esta nova versão está sendo chamada, a princípio, de brModeloNext.

4. brModeloNext

A principal motivação para o desenvolvimento da *brModeloNext* é torná-la uma ferramenta multiplataforma, ou seja, fazer com que possa ser executada independentemente do sistema operacional utilizado. Além disso, visa-se a correção das limitações da versão atual. .

Para o desenvolvimento da *brModeloNext*, optamos pela utilização da linguagem de programação Java. Dentre os diversos motivos da escolha, o principal é que Java apresenta a portabilidade como uma das suas fortes características. Tudo o que se desenvolve em Java pode ser executado em qualquer sistema operacional que possua uma máquina virtual Java. Outras vantagens levadas em consideração são: (i) disponibilidade de uma vasta quantidade de API's, ferramentas, *frameworks* e bibliotecas *open-source* de alta qualidade; (ii) alto nível de suporte através de materiais e comunidades de pesquisa, e (iii) a experiência adquirida nesta linguagem ao longo do curso de Sistemas de Informação pelos autores deste trabalho.

A idéia principal da *brModelo* é permitir que o usuário possa desenhar modelagens de BDs conceituais e lógicas no software, de uma maneira amigável. A modelagem é composta por vários elementos que representam as entidades, relacionamentos e outros conceitos de modelagem e cada um deles é representado graficamente de uma forma diferente. Essa forma de representação das modelagens conceituais e lógicas apresenta semelhanças com a representação de grafos, onde a edição é feita a partir de vértices interligados entre si através de arestas. Com o objetivo de facilitar o desenvolvimento e tirar proveito da existência de inúmeras API's para as mais diversas funções em Java, optou-se pelo uso de componentes que possam contribuir com a renderização destas representações gráficas para as modelagens.

O componente que melhor atendeu nossas expectativas e necessidades foi o JGraph [9]. JGraph é uma simples e poderosa biblioteca de renderização de grafos escrita em Java, com total compatibilidade com o componente Swing – API para desenvolvimento de interfaces gráficas utilizada em Java - , sendo utilizado por grandes corporações comerciais e instituições acadêmicas de todo o mundo para aplicações das mais diversas áreas.

O JGraph possibilita a visualização, interação do usuário, ajustes automáticos de *layout* e análises de performances com os grafos. O principal atrativo nesta API é a possibilidade de fazer adaptações para o modelo de negócio desejado – a modelagem de dados - e a facilidade com que isso pode ser feito, dada a semelhança de representação entre grafos e modelagens de dados, conforme podemos notar na Figura 4. Grafos são compostos de vértices interligados por arestas enquanto uma modelagem ER é representada por objetos como entidades e relacionamentos, interligados entre si.

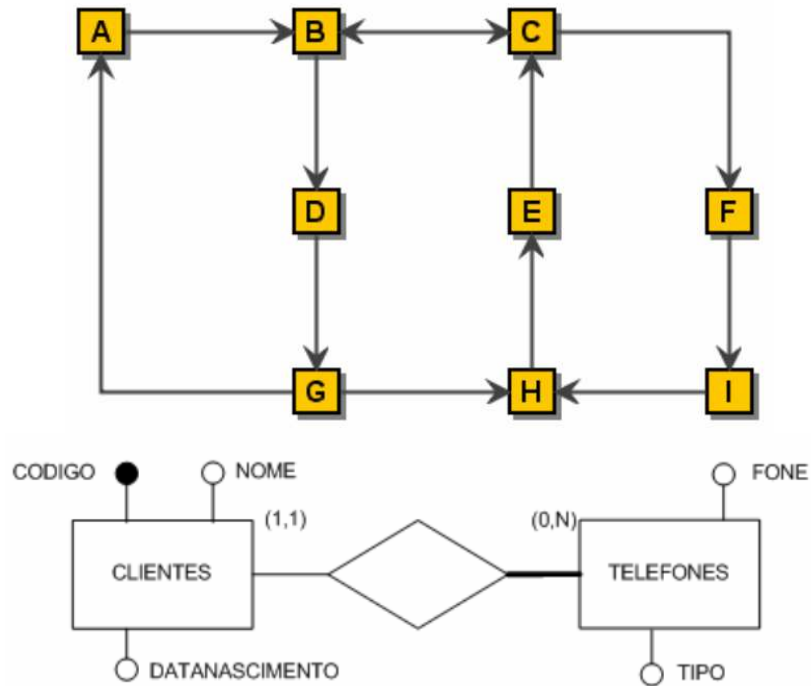


Figura 4: Semelhança entre a representação de grafos (acima) e a representação de modelagem conceitual de dados (abaixo)

4.1 Funcionalidades

A elaboração da interface gráfica da brModeloNext levou em conta a interface da brModelo, propondo algumas melhorias quanto à usabilidade da ferramenta (ver Seção 4.2). O JGraph, componente base utilizado no desenvolvimento da ferramenta, possui ampla gama de ferramentas e funcionalidades para formatação de componentes visuais. Entretanto, várias dessas ferramentas não agregam funcionalidades interessantes à modelagem de BD. Sendo assim, optou-se por uma readequação das ferramentas mais facilmente acessadas pelo usuário, a fim de tornar o uso mais prático e intuitivo, permitindo inclusive a melhor visualização dos componentes na tela de edição, conforme ilustrado na Figura 5.

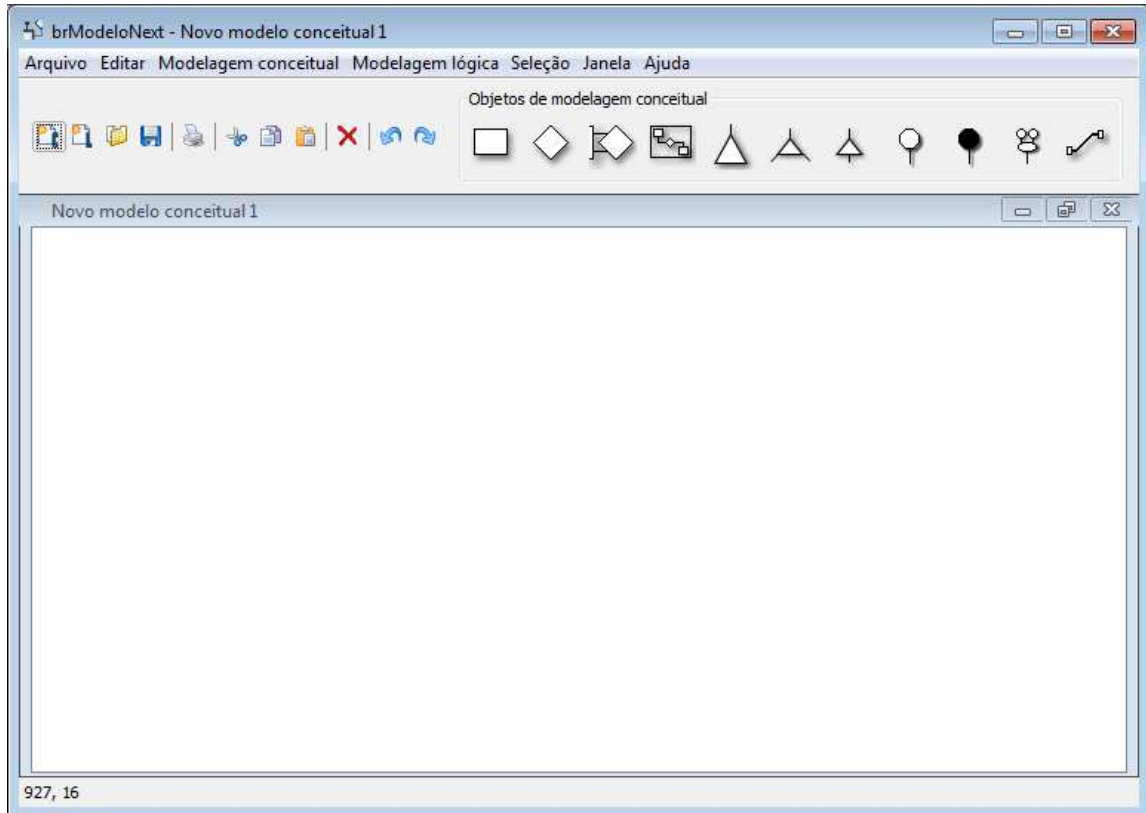


Figura 5: Tela principal da brModeloNext

A barra de ferramentas incluída na brModeloNext segue o padrão usual de outras ferramentas de edição de conteúdo. Desta forma, aproveita-se a experiência do usuário, inclusive no reconhecimento dos ícones. Esta barra de ferramentas disponibiliza ao usuário as funcionalidades de criação de novas modelagens (conceituais e lógicas), abertura de projetos anteriormente criados, gravação, funcionalidades de área de transferências (recortar, copiar e colar), além das funções de desfazer e refazer alterações realizadas nas modelagens.

No lado direito da barra de ferramentas estão as opções de edição de modelagem conceitual e lógica, que disponibilizam a paleta de objetos a serem incluídos no modelo. Esta barra de ferramentas é sensível ao contexto da modelagem, ou seja, é exibida de acordo com o tipo de modelo que está sendo construído ou alterado pelo usuário, a fim de evitar que o usuário utilize componentes inadequados no seu projeto. Para uma modelagem conceitual, as barras de ferramentas e de objetos de modelagem apresentam-se conforme a Figura 5.

A barra de objetos de modelagem conceitual disponibiliza os principais conceitos do modelo ER: entidades; relacionamentos; auto-relacionamento; entidade associativa, especialização, especialização exclusiva, especialização não-exclusiva, atributo, atributo identificador, atributo composto e conector. Um simples clique na opção e um clique seguinte na posição da tela onde o usuário deseja a inserção do objeto, implica na criação do novo componente.

O posicionamento da barra de objetos de modelagem foi alterado em relação à brModelo. Desta forma, disponibiliza-se ao usuário maior área de edição para o projeto, sendo essa uma das melhorias sugeridas.

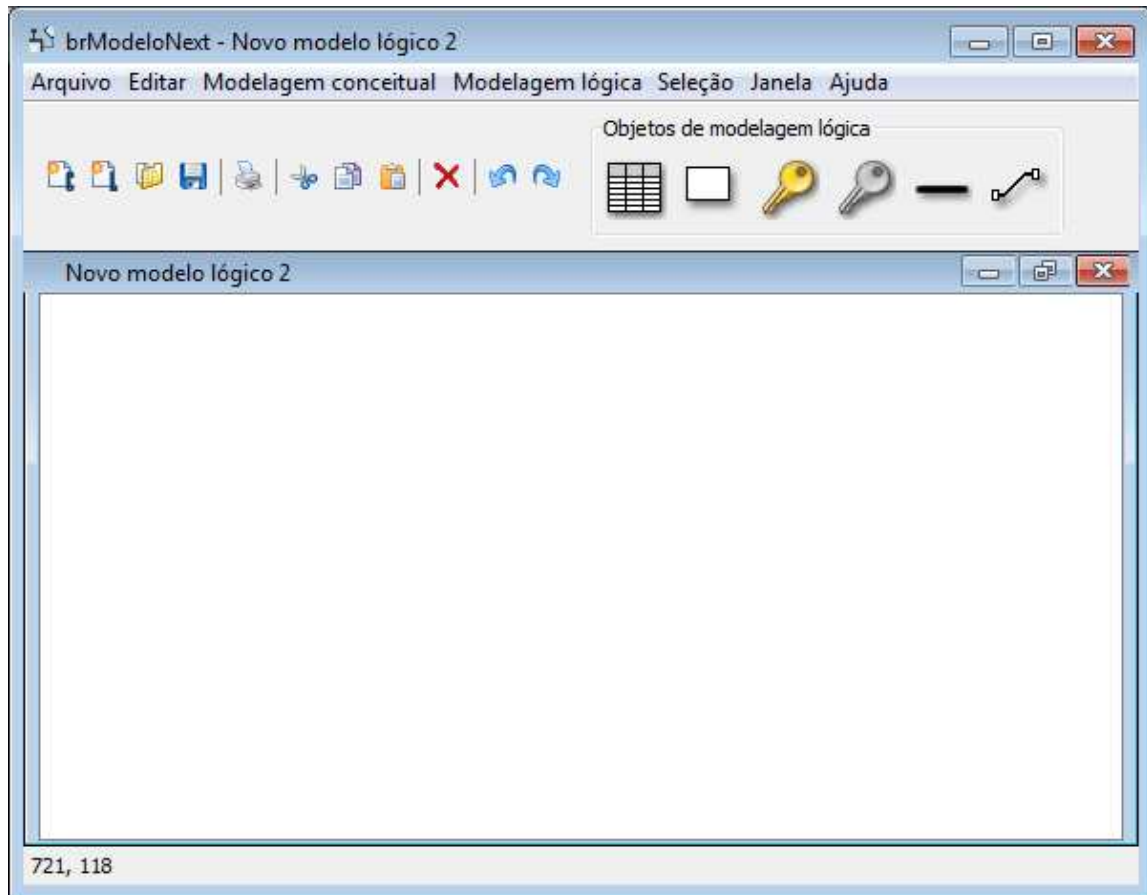


Figura 6: Tela para uma modelagem lógica na brModeloNext

A interface da brModeloNext para atividades de modelagem lógica é mostrada na Figura 6. Na barra de objetos têm-se as seguintes opções: tabela campo chave primária chave estrangeira separador de atributos e conector. De forma similar à barra de objetos de modelagem conceitual, com um clique no objeto da barra e outro no projeto, tem-se a inclusão do item selecionado.

Há também algumas funcionalidades intermediárias, sejam elas de validação ou suporte ao usuário, no que se refere à inclusão dos componentes no projeto. Estas funcionalidades seguem regras de modelagem, evitando inclusões ou alterações não são permitidas em determinados contextos. Para tanto, a brModeloNext lança mão de três vias de funcionalidade: i) descrição contextual, através da posição da seta do mouse, sobre a barra de ferramentas – funcionalidade conhecida como *tooltip*, indicando a classe do objeto selecionado (ver Figura 7); ii) descrição, na barra de status, do próximo passo a ser seguido pelo usuário para a efetiva inclusão do seu objeto no projeto (ver Figura 8); iii) validação no momento da tentativa de inclusão ou alteração no projeto, emitindo mensagem de erro ao usuário, informando-lhe o motivo da recusa da inclusão (ver Figura 9).

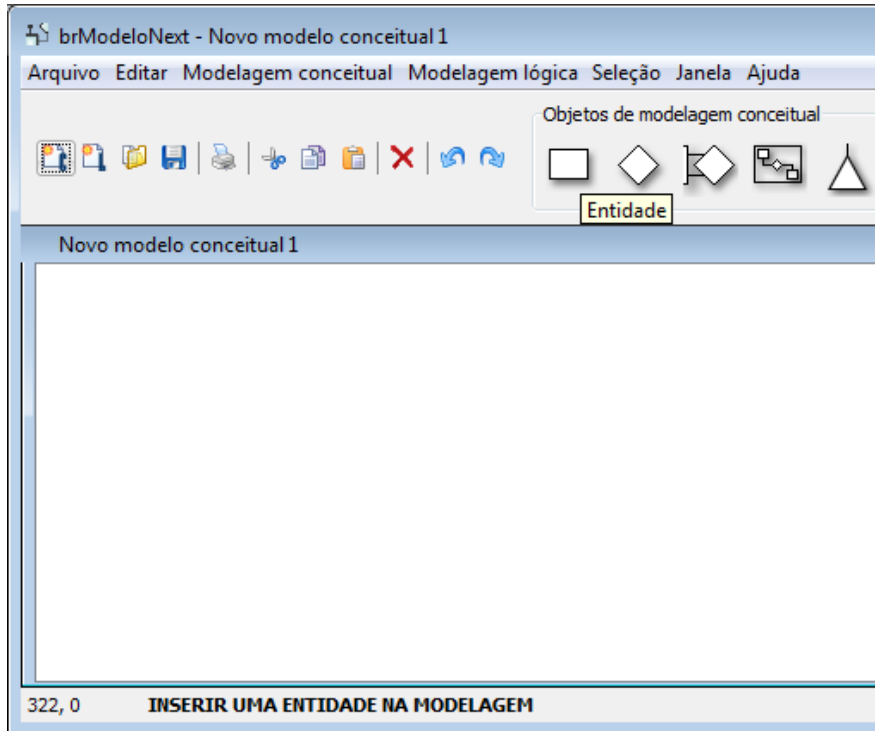


Figura 7: Informações sobre o objeto através de tooltip e barra de status

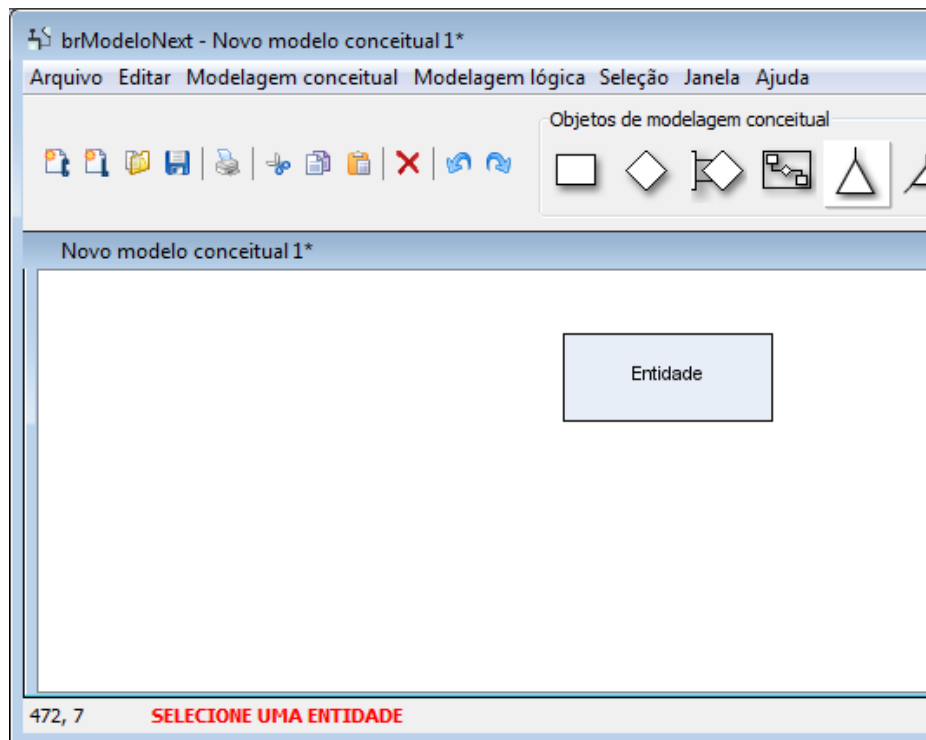


Figura 8: Descrição destacada do passo a ser executado para a inserção do objeto no modelo através da barra de status

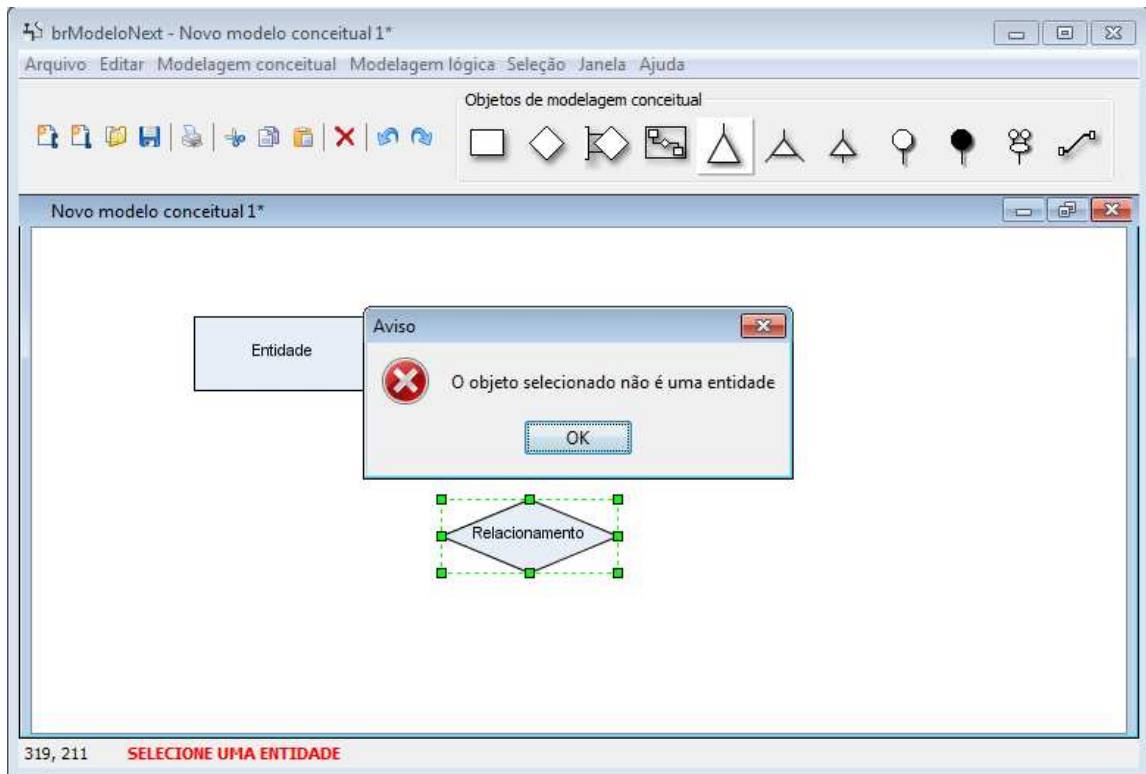


Figura 9: Exemplo de uma tentativa de incluir um objeto de forma equivocada na modelagem e o aviso alertando o erro cometido pelo usuário

A brModeloNext permite ao usuário a manipulação de diversos projetos ao mesmo tempo. Tal possibilidade foi criada a partir da implementação do padrão MDI – *Multiple Documents Interface* – permitindo que o usuário manipule várias janelas de projetos, internamente à janela principal do sistema, conforme mostrado na Figura 10. Esta funcionalidade visa suprir uma das carências da versão anterior, que permite a visualização de apenas um projeto em sua execução. Como a principal funcionalidade da ferramenta é a conversão entre modelos conceituais e lógicos, trabalhar com pelo menos dois modelagens simultaneamente facilita a validação do projeto como um todo pelo usuário.

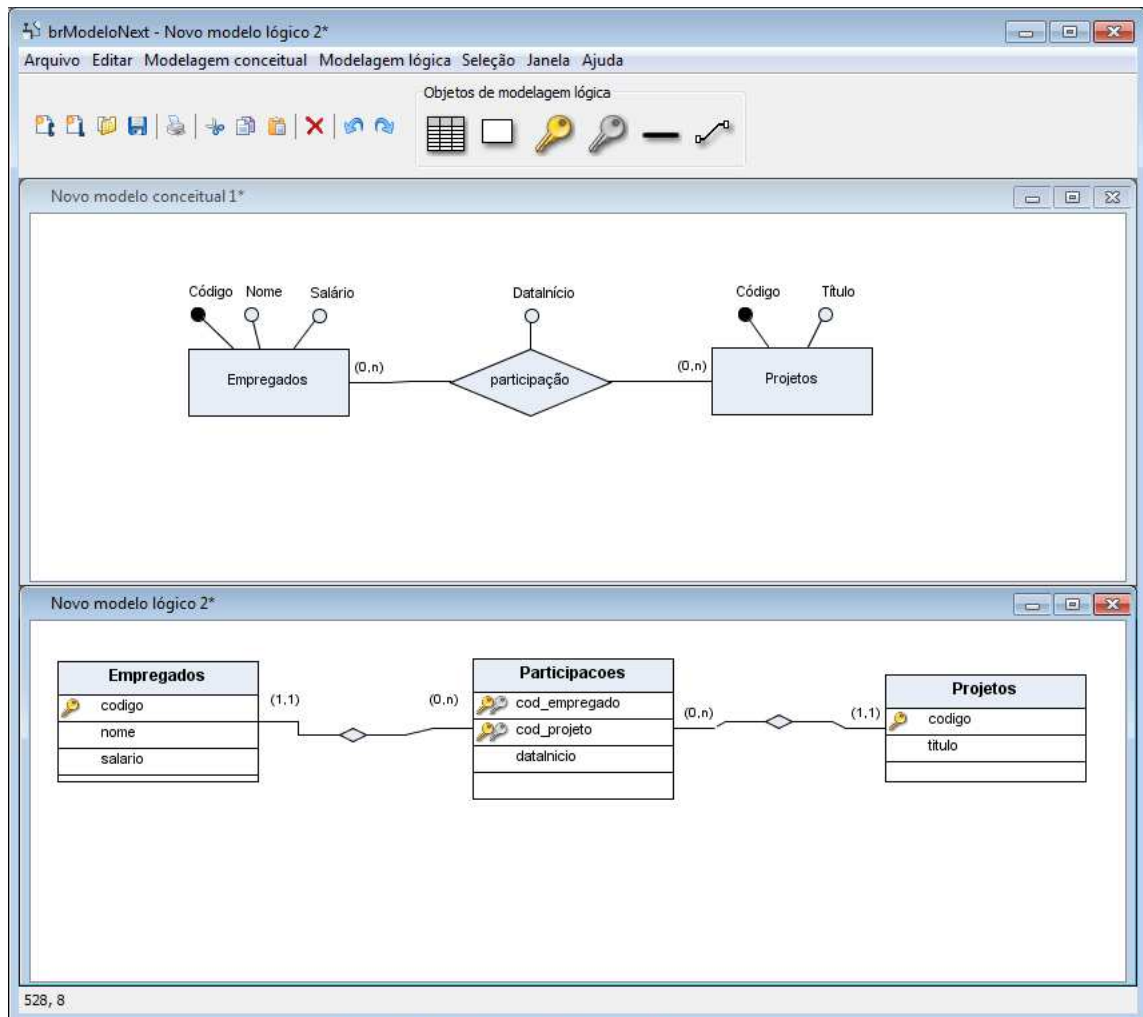


Figura 10: Visualização de 2 modelagens na mesma tela

A Barra de menu da brModeloNext foi remodelada para seguir três princípios, visando tornar seu uso mais intuitivo. Estes princípios são:

- i) Exibir ao usuário apenas as opções que lhe são permitidas, inclusive no que diz respeito ao contexto de operação, ou seja, ao tipo de modelagem que está sendo efetuada;
- ii) Utilizar a mesma nomenclatura e chamadas das *tooltips* da barra de ferramentas, tornando a interface mais uniforme e mais fácil de compreender;
- iii) Ênfase nas modelagens, em detrimento à edição e formatação, conduzindo o foco do usuário aos projetos conceituais e lógicos.

Entretanto, aproveitando-se a experiência do usuário em ferramentas de edição de conteúdo, dá-se ao usuário a possibilidade de acessar e manipular, a partir do menu, funcionalidades de arquivo padrão, como gravação e abertura dos mesmos; navegação entre as janelas de edição; acesso aos arquivos de ajuda do programa.

Através dos itens de menu nomeados “Modelagem Conceitual” e “Modelagem Lógica”, o usuário pode incluir todos os respectivos objetos de classes de modelagem no

seu projeto – da mesma forma que na barra de ferramentas –, bem como efetuar a conversão entre tipos de modelagem, como exibido nas Figuras 11 e 12.



Figura 11: Menu “Modelagem conceitual”

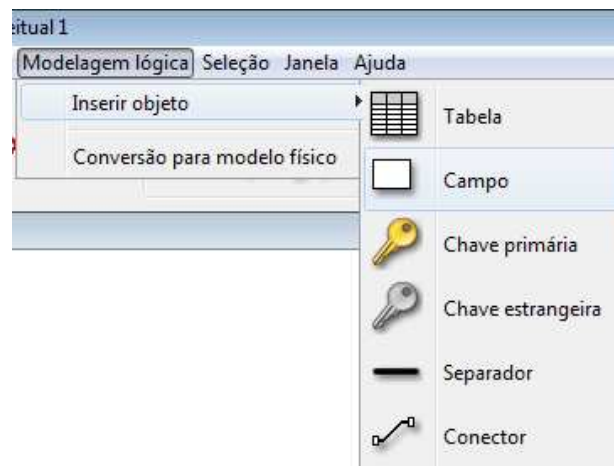


Figura 12: Menu “Modelagem lógica”

Partindo do princípio que a área de edição do projeto deve ser a maior possível para facilitar o processo de projeto, além do deslocamento das funções principais do sistema para a barra de ferramentas e de menu, lançou-se mão da edição “*in-place*”, ou seja, janelas que são exibidas em tempo de execução no momento da criação de cada objeto, ou quando o usuário assim necessitar, através de um duplo-clique no objeto a ser editado, possibilitando a edição das respectivas propriedades, conforme a Figura 13. A versão anterior possuía uma janela lateral que exibia e permitia a alteração destas propriedades. Entretanto, durante a edição da modelagem – e não de cada objeto propriamente dito – esta janela tomava um espaço valioso da janela, especialmente se

considerarmos uma tela de proporção 3:4, em que manipuladores de imagem já têm seu espaço de edição bastante prejudicado. Tal qual as funcionalidades de menu anteriores, esta também se aproveita de experiências anteriores do usuário para tornar o uso da brModeloNext mais intuitivo.

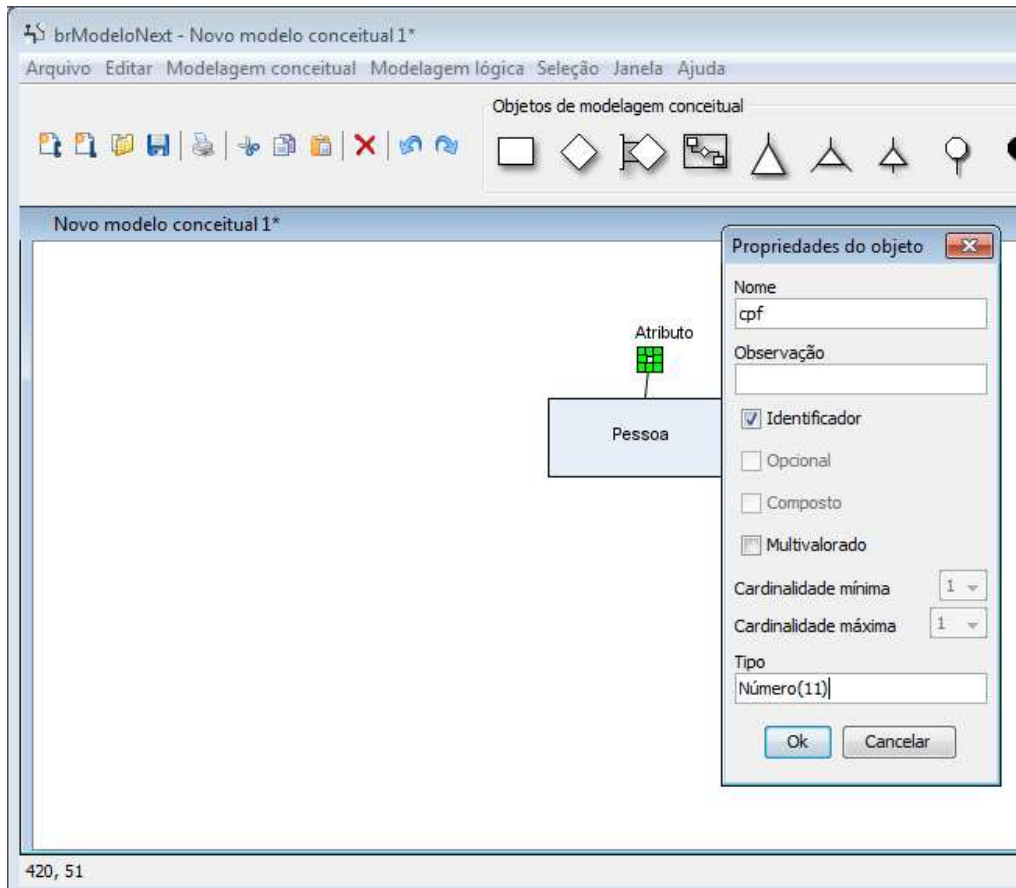


Figura 13: Janela de edição *in-place*, acionada ao inserir o objeto na modelagem e também ao realizar um duplo-clique sobre o objeto

Os mesmos valores que podem ser editados nesta janela *in-place* podem ser visualizados em uma *tooltip*, ao fixar o ponteiro do mouse sobre o objeto por alguns instantes, conforme mostra a Figura 14.

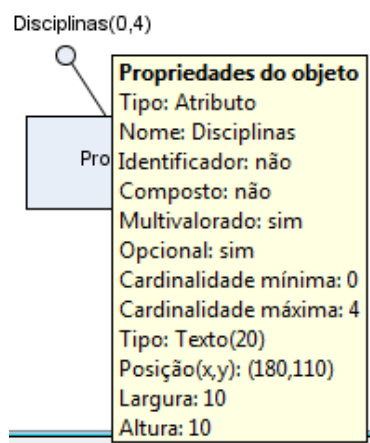


Figura 14: Descrição detalhada do objeto exibida em uma *tooltip* ao fixar o ponteiro do mouse no objeto

No processo de conversão do modelo conceitual para o modelo lógico, na etapa de mapeamento das especializações e na etapa de mapeamento dos relacionamentos e seus atributos, a ferramenta auxilia o usuário interagindo através de janelas que questionam qual das alternativas de mapeamento ela deve escolher, conforme as Figuras 15 e 16.

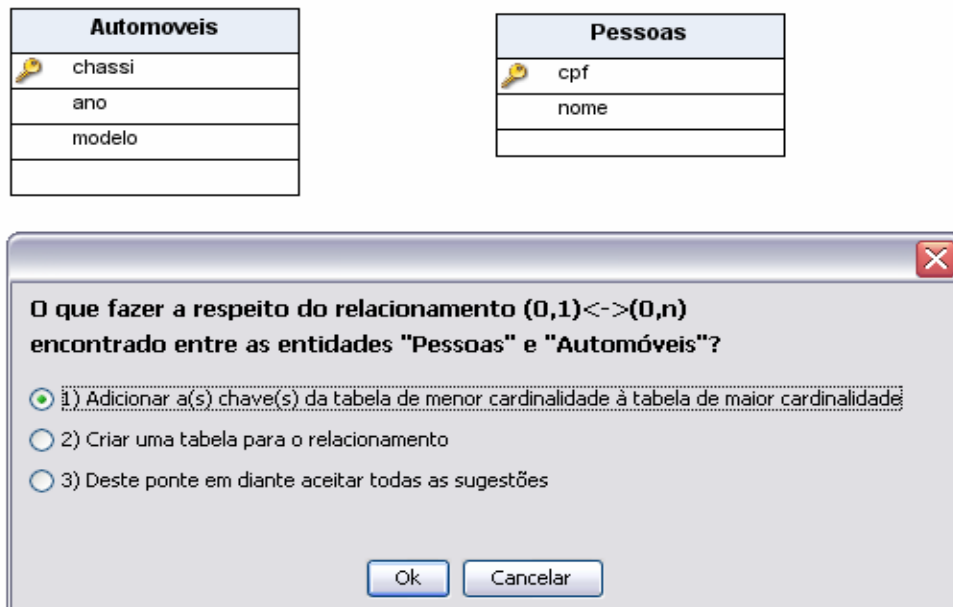


Figura 15: Exemplo de interatividade entre a brModeloNext e o usuário no processo de mapeamento de um relacionamento e seus atributos

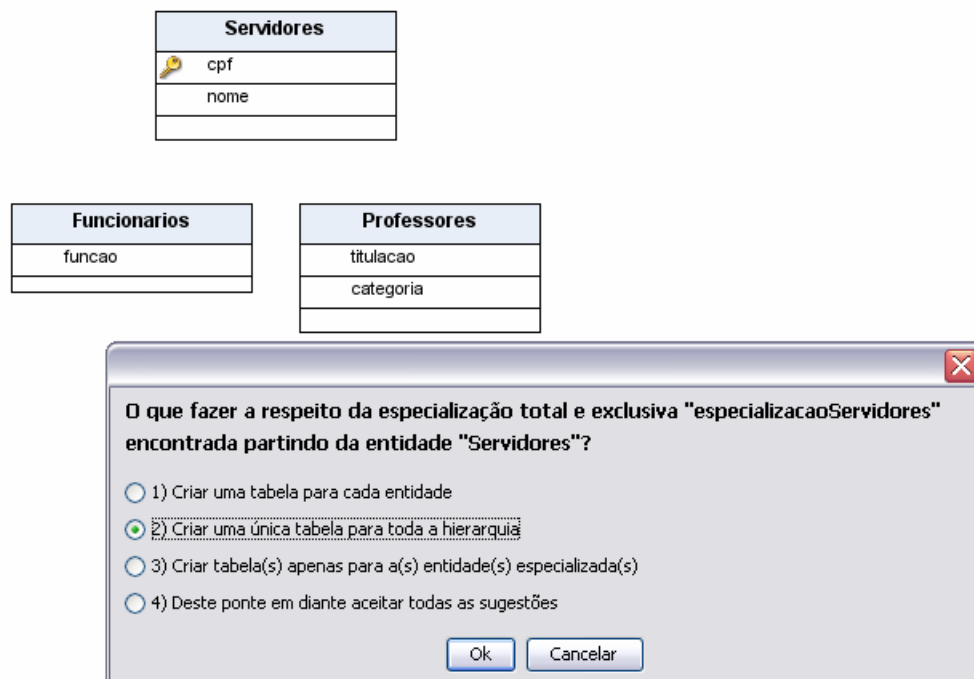


Figura 16: Exemplo de interatividade entre a brModeloNext e o usuário no processo de mapeamento de uma especialização

Das idéias principais da nova versão, nenhuma permaneceu mais pujante que a portabilidade entre sistemas operacionais. Apesar da utilização de componentes e bibliotecas completamente portáveis, a forma de renderização dos componentes é

diferente em cada sistema operacional. Na realidade, dois ambientes de usuário de um mesmo sistema operacional já podem ter uma sensível diferença na renderização do sistema.

Assim sendo, disponibilizou-se a seleção do estilo de ambiente, conhecida como *“look-and-feel”* (funcionalidade disponibilizada pelo componente de criação de interfaces gráficas da tecnologia Java chamado Swing) que poderia – e até deveria - ter sido suprimida do projeto para o aumento de sua portabilidade, evitando que o usuário estranhe a interface ao migrar de um sistema operacional para outro. Porém, por tratar-se de um editor gráfico em sua essência, esta funcionalidade tornou-se salutar. A princípio, o sistema é disponibilizado ao usuário com esta propriedade configurada de acordo com o seu sistema operacional. Entretanto, ele ainda tem outras possibilidades de aparências, que não interferem em nada nas funcionalidades e na usabilidade da ferramenta, e são completamente portáveis, ao contrário das nativas de seus ambientes.

4.2 Análise e Planejamento da Interface com o Usuário da brModelo

Uma das premissas do desenvolvimento da brModeloNext é a melhoria da usabilidade do sistema. Para tanto, faz-se necessária uma análise da interface atual com o usuário e planejamento das melhorias sobre ela. Assim, a análise de melhorias é focada no usuário e nas tarefas por ele executadas. Ambientes físicos, técnicos e organizacionais não se aplicam ao caso, já que os ambientes de execução da ferramenta são completamente heterogêneos.

A fim de manter o foco do desenvolvimento no usuário, visando manter e criar tarefas as mais intuitivas possíveis, o projeto da interface foi baseado nas seguintes etapas [10]:

i) Definição do Escopo da Ferramenta

Definir o escopo de utilização da ferramenta deve ser sempre o ponto de partida do desenvolvimento de um software. Este escopo será a base para todo o restante do processo, e é um risco ignorá-lo, sob pena de haver significativa fuga do foco do desenvolvimento e conseqüente perda das características principais de um software. É necessário definir, nesta fase, os requisitos funcionais e não-funcionais do sistema ou ferramenta.

Para a brModeloNext, esta etapa apenas reafirma as funcionalidades já existentes na brModelo, aplicando sobre elas o seu diferencial: melhoria de usabilidade e portabilidade, ambos requisitos não funcionais da ferramenta.

ii) Análise do Trabalho

Os resultados da análise de trabalho com a brModelo balizam as modificações para geração da interface com o usuário do brModeloNext. Tal abordagem visa garantir que usuários da brModelo consigam se adaptar com facilidade à forma de interação da

brModeloNext, de forma a atingir as suas reais expectativas. Além disso, evita-se repulsa primária do usuário com relação à mudança de abordagem, versão ou nova ferramenta.

Em geral, a análise de trabalho prevê atividades de coleta de informação, entrevistas e observações, além de análise de ambiente. Entretanto, dado o foco da ferramenta e por ser um escopo completamente fechado, a análise de trabalho baseou-se tão somente em coletas de informações e observações de uso dos próprios desenvolvedores, fazendo esta análise como balizamento paralelo ao desenvolvimento da brModeloNext.

4.3 Modelagem da Solução

A ferramenta brModeloNext foi desenvolvida aplicando padrões de projetos orientados a objeto que permitem o desacoplamento das responsabilidades de cada artefato dentro do software. Além disso, a arquitetura do software feita em camadas facilita a manutenção e permite que extensões e novas funcionalidades sejam implementadas sem impacto nas características anteriores.

A arquitetura em camadas foi modelada a partir do modelo MVC (Model-View-Controller), onde cada camada possui pacotes e classes com responsabilidades específicas.

Responsável por controlar as regras de negócio de relacionamentos entre objetos e a conversão entre modelos, estão as classes do pacote Controller, conforme a figura 17.

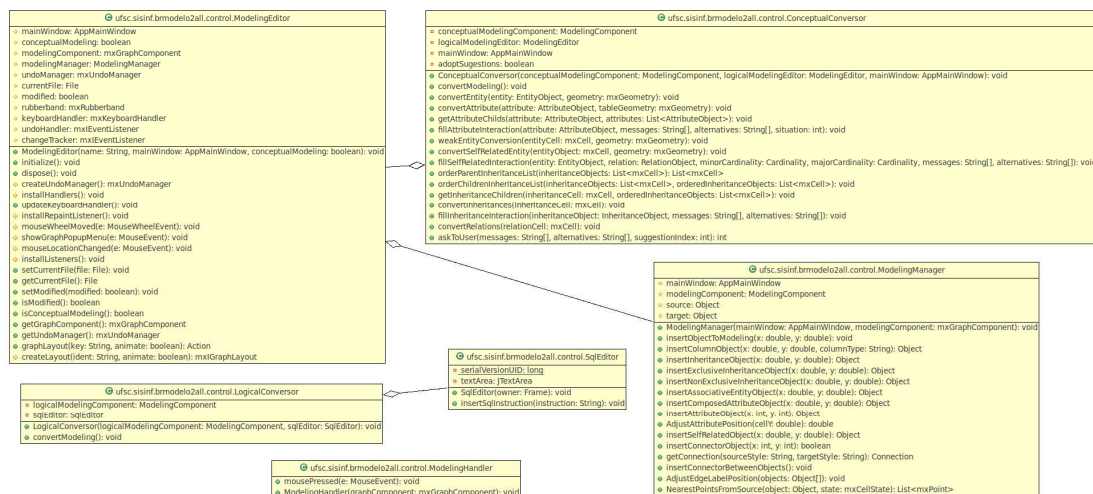


Figura 17: Modelagem das classes de controle da BrModeloNext

Os shapes, que representam os objetos da modelagem em tela, são representados em classes de entidades, conforme as figuras 18 e 19, abaixo. A modelagem foi fragmentada para melhor visualização.

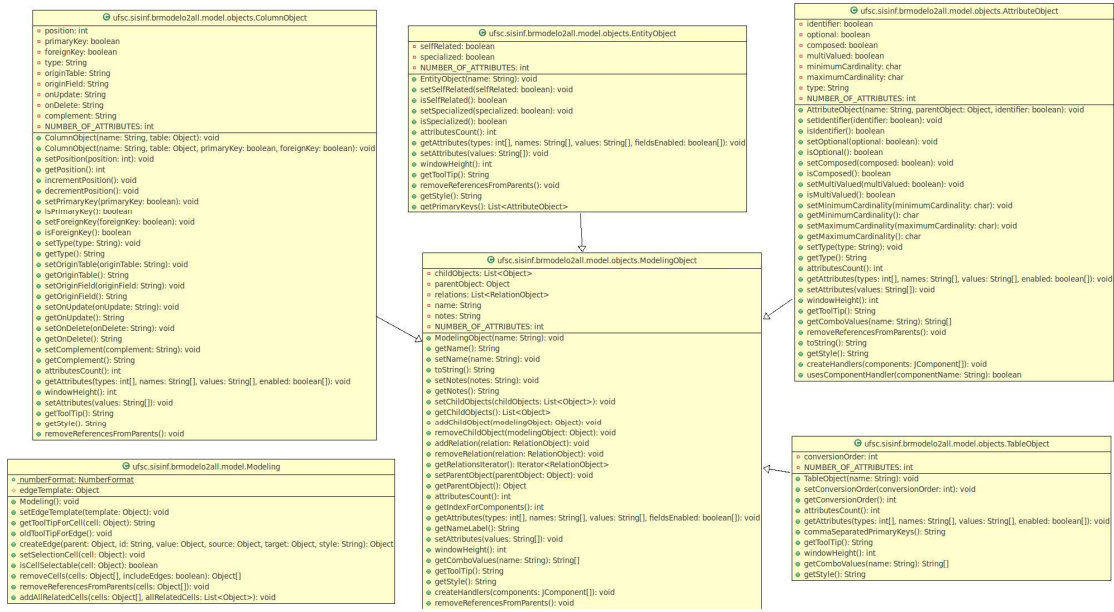


Figura 18: Modelagem de classes de entidade da BrModeloNext

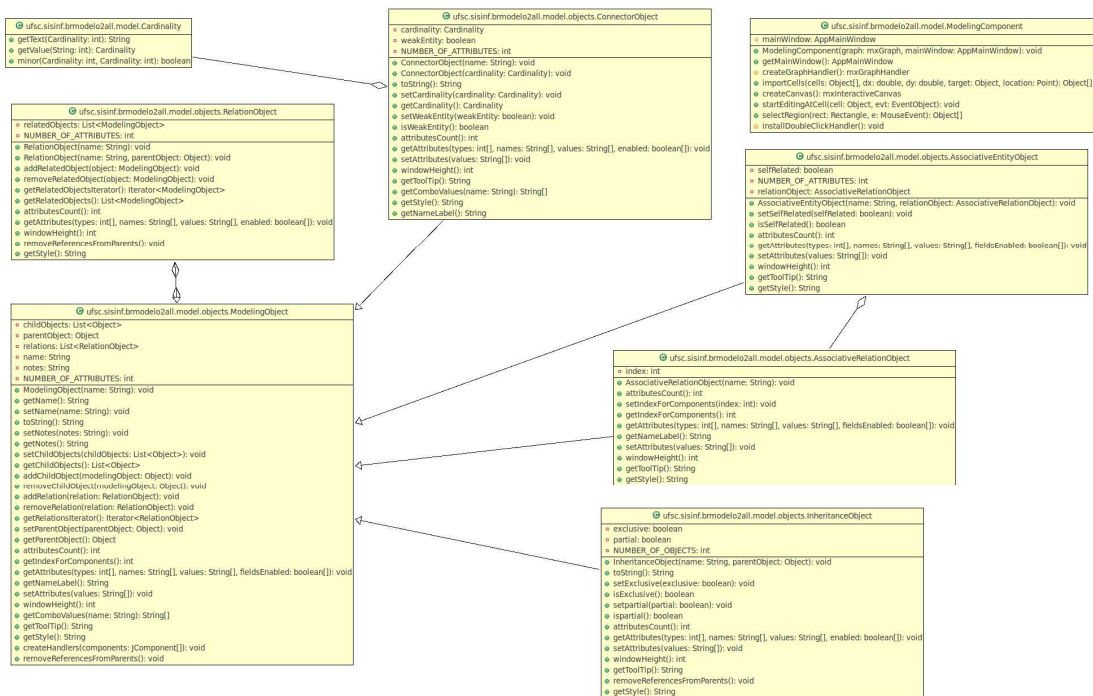


Figura 19: Modelagem de classes de entidade da BrModeloNext

As classes foram modeladas e implementadas seguindo-se as convenções de código Oracle, responsável pela linguagem Java, conforme demonstram as figuras acima. Desta forma facilita-se o entendimento da solução para futuros desenvolvedores da solução.

5. Conclusão

O desenvolvimento de qualquer sistema informatizado deve sempre vislumbrar melhorias futuras, visando torná-lo cada vez mais atrativo ao usuário que necessita manipulá-lo. A brModelo é uma ferramenta que já tem grande aceitação no meio acadêmico brasileiro, principalmente no ensino de modelagem de BD em disciplinas da área. Esta popularidade da ferramenta, aliada ao *feedback* dado por diversos usuários com relação a sugestões de melhoria e necessidade de portabilidade motivaram o desenvolvimento da brModeloNext.

A brModelo surgiu como uma alternativa à outras ferramentas case disponíveis para o projeto de BDs relacionais. Em sua maioria, as boas ferramentas CASE são tecnologias proprietárias. Algumas ferramentas gratuitas estão disponíveis, até mesmo softwares livres, mas seu foco é mercadológico, ficando o aspecto acadêmico em segundo plano. A brModelo veio preencher estas lacunas: uma ferramenta gratuita, de código aberto e que mantém seu foco no desenvolvimento do projeto de BD desde o início, ou seja, desde a modelagem conceitual. Esta etapa de projeto normalmente é ignorada pelas ferramentas CASE disponíveis no mercado, que iniciam pela modelagem lógica em suas abordagens.

Em todos estes aspectos a brModelo obteve êxito. Entretanto, ela possui limitações, seja com relação a funcionalidades ou tecnologia. A principal delas reside no fato de ter sido desenvolvida em Delphi / Object Pascal, o que faz com que ela só possa ser executada nativamente em sistemas operacionais Windows. Partindo do princípio que o foco da utilização maior da ferramenta é o meio acadêmico, aliado ao fato de que há um grande movimento em prol da adoção de tecnologias baseadas em software livre em diversos meios, cresceu com o tempo a necessidade de tornar essa ferramenta portátil, para que se pudesse utilizá-la em quaisquer sistemas operacionais.

Desta necessidade nasceu a brModeloNext, aproveitando o ótimo legado da brModelo. Todas as funções de modelagem conceitual e lógica, bem como os mapeamentos já existentes na brModelo foram reimplementadas na brModeloNext, adicionando-se melhorias de usabilidade como a edição in-place, as tooltips descritivas e uma melhor sensibilidade das ferramentas quanto ao contexto da sua aplicação como na promoção de atributos em entidades, por exemplo.

Outra contribuição da brModeloNext é a correção de diversos erros ou limitações existentes na brModelo, que vinham sendo reportados pelos usuários.

- Correções na modelagem conceitual:
 - não é possível promover um auto-relacionamento para entidade associativa;

- não é possível especializar uma entidade associativa;
- uma entidade associativa deve estar conectada a pelo menos um relacionamento, ou seja, ela não pode ter conexões apenas com o relacionamento dentro dela;
- Correções na modelagem lógica:
 - chaves estrangeiras às vezes são posicionadas erradas quando ocorre o mapeamento de um relacionamento 1-1 ou 1-N;
 - quando uma tabela é gerada para um relacionamento M-N e o mapeamento precisa fazer relacionar uma table em outra através de chave estrangeira mas esta também é uma chave primária, a brModelo não utiliza a notação gráfica para atributo que é chave primária e chave estrangeira ao mesmo tempo;
 - não está sendo respeitada a regra de conversão de uma entidade fraca que consiste em adicionar a chave primária da entidade forte como parte da chave primária na tabela criada para a entidade fraca;
 - cria campos a mais do que deveria ao realizar mapeamento de atributos multivalorados

5.1 Trabalhos Futuros

A brModeloNext trouxe contribuições para atividade de projeto de BDs relacionais, corrigindo e aprimorando a ferramenta brModelo. Entretanto, como projeto de código aberto, vislumbra-se atividades futuras para a sua melhoria contínua.

Dentre os diversos e possíveis trabalhos futuros, destaca-se:

i) realizar testes in-loco com os usuários da ferramenta, a fim de conhecer oportunidades de melhoria de sua usabilidade para futura implementação, bem como corrigir problemas inerentes a esta primeira versão recém-implementada;

ii) disponibilizar a ferramenta em ambiente Web, visando aumentar ainda mais a portabilidade da mesma;

iii) traduzir a ferramenta para outros idiomas, tornando-a de uso mais abrangente;

iv) especificar, redigir e disponibilizar documentação de ajuda, seja por meio de tutoriais, wiki ou similares, seja com relação à utilização do sistema ou a teoria associada à mesma.

6. Referências Bibliográficas

- [1] KORT, Henry F. & SILBERSCHATZ, Abraham. Sistemas de bancos de dados. 2. ed. São Paulo: Makron Books, 1995.
- [2] HEUSER, Carlos Alberto. Projeto de banco de dados. 6. ed. Porto Alegre: Instituto de informática da UFRGS, Sagra Luzzato, 2009.
- [3] Computer Associates ;ERwin® Data Modeler. Disponível na internet no endereço: <http://erwin.com> Acesso em: Novembro de 2010.
- [4] IBM; Rational Rose. Disponível na internet no endereço: <http://www-142.ibm.com/software/products/br/pt/datamodeler/> Acesso em: Novembro de 2010.
- [5] FabForce ; DBDesigner. Disponível na internet no endereço: <http://fabforce.net/dbdesigner4/> Acesso em: Novembro de 2010.
- [6] Grupo de banco de dados – Universidade Federal de Santa Catarina. Disponível na internet no endereço: <http://www.gbd.inf.ufsc.br> Acesso em: Novembro de 2010.
- [7] brModelo. Disponível na internet no endereço: <http://www.sis4.com/brModelo/> Acesso em: Agosto de 2010.
- [8] DEITEL, H. M. & DEITEL, P. J. Java como programar. 4ª ed. Porto Alegre: Bookman, 2003.
- [9] JGraph Ltda.; JGraph. Disponível na internet no endereço: <http://www.jgraph.com/> Acesso em Setembro de 2010.
- [10] CYBIS, Walter de Abreu. Engenharia de Usabilidade: uma abordagem ergonômica. Florianópolis: Laboratório de Utilizabilidade de Informática, UFSC, 2003.

7. Apêndices

7.1 Código-fonte

```
package ufsc.sisinf.brmodelo2all.app;

import javax.swing.UIManager;
import ufsc.sisinf.brmodelo2all.ui.AppMainWindow;
import ufsc.sisinf.brmodelo2all.ui.MenuBar;

public class BrModelo2All {

    public static void main(String[] args) {
        try {
            UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (Exception e1) {
            e1.printStackTrace();
        }
        AppMainWindow app = new AppMainWindow();
        app.createFrame(new MenuBar(app)).setVisible(true);
    }
}
```

```
package ufsc.sisinf.brmodelo2all.control;

import java.awt.event.MouseEvent;

import ufsc.sisinf.brmodelo2all.model.objects.ModelingObject;
import ufsc.sisinf.brmodelo2all.model.ModelingComponent;

import com.mxgraph.model.mxCell;
import com.mxgraph.swing.mxGraphComponent;
import com.mxgraph.swing.handler.mxGraphHandler;

public class ModelingHandler extends mxGraphHandler {

    public ModelingHandler(mxGraphComponent graphComponent) {
        super(graphComponent);
    }

    @Override
    public void mousePressed(MouseEvent e) {

        Object cell = graphComponent.getCellAt(e.getX(), e.getY(), false);

        if (cell != null && ((mxCell)cell).getValue() instanceof ModelingObject)
            ((ModelingComponent)graphComponent).
                getMainWindow().updateSelectionMenu((ModelingObject)((mxCell)cell).getValue());
        else

            ((ModelingComponent)graphComponent).getMainWindow().updateSelectionMenu(null);

        super.mousePressed(e);
    }
}
```

```
package ufsc.sisinf.brmodelo2all.control;

import com.mxgraph.model.mxCell;
import ufsc.sisinf.brmodelo2all.model.ModelingComponent;
```



```

import ufsc.sisinf.brmodelo2all.model.objects.TableObject;

public class LogicalConversor {

    private final ModelingComponent logicalModelingComponent;

    private final SqlEditor sqlEditor;

    public LogicalConversor(final ModelingComponent logicalModelingComponent, final SqlEditor
sqlEditor) {
        this.logicalModelingComponent = logicalModelingComponent;
        this.sqlEditor = sqlEditor;
    }

    public void convertModeling() {
        Object[] cells =
logicalModelingComponent.getCells(logicalModelingComponent.getBounds());
        for (int i = 0; i < cells.length; i++) {
            Object cell = cells[i];
            if (cell instanceof mxCell) {
                mxCell objectCell = (mxCell)cell;
                if (objectCell.getValue() instanceof TableObject) {
                    TableObject tableObject = (TableObject) objectCell.getValue();
                    sqlEditor.insertSqlInstruction("CREATE TABLE " +
tableObject.getName() + "()");
                }
            }
        }
    }
}

```

```

package ufsc.sisinf.brmodelo2all.control;

```

```

import java.awt.Color;
import java.awt.Point;
import java.awt.event.ActionEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.MouseMotionListener;
import java.awt.event.MouseWheelEvent;
import java.awt.event.MouseWheelListener;
import java.beans.PropertyVetoException;
import java.io.File;
import java.util.List;

import javax.swing.AbstractAction;
import javax.swing.Action;
import javax.swing.JInternalFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.SwingUtilities;
import javax.swing.event.InternalFrameEvent;
import javax.swing.event.InternalFrameListener;

import ufsc.sisinf.brmodelo2all.model.Modeling;
import ufsc.sisinf.brmodelo2all.model.ModelingComponent;
import ufsc.sisinf.brmodelo2all.model.objects.ModelingObject;
import ufsc.sisinf.brmodelo2all.ui.AppMain Window;
import ufsc.sisinf.brmodelo2all.ui.KeyboardHandler;
import ufsc.sisinf.brmodelo2all.ui.PopupMenu;
import com.mxgraph.layout.mxCircleLayout;

```

```

import com.mxgraph.layout.mxCompactTreeLayout;
import com.mxgraph.layout.mxEdgeLabelLayout;
import com.mxgraph.layout.mxIGraphLayout;
import com.mxgraph.layout.mxOrganicLayout;
import com.mxgraph.layout.mxParallelEdgeLayout;
import com.mxgraph.layout.mxPartitionLayout;
import com.mxgraph.layout.mxStackLayout;
import com.mxgraph.layout.hierarchical.mxHierarchicalLayout;
import com.mxgraph.model.mxCell;
import com.mxgraph.swing.mxGraphComponent;
import com.mxgraph.swing.mxGraphOutline;
import com.mxgraph.swing.handler.mxKeyboardHandler;
import com.mxgraph.swing.handler.mxRubberband;
import com.mxgraph.swing.util.mxMorphing;
import com.mxgraph.util.mxEvent;
import com.mxgraph.util.mxEventObject;
import com.mxgraph.util.mxRectangle;
import com.mxgraph.util.mxResources;
import com.mxgraph.util.mxUndoManager;
import com.mxgraph.util.mxUndoableEdit;
import com.mxgraph.util.mxEventSource.mxEventListener;
import com.mxgraph.util.mxUndoableEdit.mxUndoableChange;
import com.mxgraph.view.mxGraph;

@SuppressWarnings("serial")
public class ModelingEditor extends JFrame {

    static {
        try {
            mxResources.add("/ufsc/sisinf/brmodelo2all/ui/resources/editor");
        } catch (Exception e) {
        }
    }

    protected final AppMainWindow mainWindow;
    protected final boolean conceptualModeling;

    protected mxGraphComponent modelingComponent;
    protected ModelingManager modelingManager;
    protected mxUndoManager undoManager;
    protected File currentFile;

    protected boolean modified = false;
    protected mxRubberband rubberband;
    protected mxKeyboardHandler keyboardHandler;

    protected mxEventListener undoHandler = new mxEventListener() {
        public void invoke(Object source, mxEventObject evt) {
            undoManager.undoableEditHappened((mxUndoableEdit) evt
                .getProperty("edit"));
        }
    };

    protected mxEventListener changeTracker = new mxEventListener() {
        public void invoke(Object source, mxEventObject evt) {
            setModified(true);
        }
    };

    public ModelingEditor(String name, final AppMainWindow mainWindow,
        boolean conceptualModeling) {
        super(name);
        this.mainWindow = mainWindow;
        this.conceptualModeling = conceptualModeling;
    }

```

```

// Stores a reference to the graph and creates the command history
modelingComponent = new ModelingComponent(new Modeling(), mainWindow);
final mxGraph graph = modelingComponent.getGraph();
undoManager = createUndoManager();

modelingComponent.setFoldingEnabled(false);
// Do not change the scale and translation after files have been loaded
graph.setResetViewOnRootChange(false);

// Updates the modified flag if the graph model changes
graph.getModel().addListener(mxEvent.CHANGE, changeTracker);

// Adds the command history to the model and view
graph.getModel().addListener(mxEvent.UNDO, undoHandler);
graph.getView().addListener(mxEvent.UNDO, undoHandler);

// Keeps the selection in sync with the command history
mxEventListener undoHandler = new mxEventListener() {
    public void invoke(Object source, mxEventObject evt) {
        List<mxUndoableChange> changes = ((mxUndoableEdit) evt
            .getProperty("edit")).getChanges();
        graph.setSelectionCells(graph
            .getSelectionCellsForChanges(changes));
    }
};

undoManager.addListener(mxEvent.UNDO, undoHandler);
undoManager.addListener(mxEvent.REDO, undoHandler);

modelingManager = new ModelingManager(mainWindow, modelingComponent);

setClosable(true);
setMaximizable(true);
setIconifiable(true);
setResizable(true);
setBounds(20 * AppMainWindow.windowCount,
    20 * AppMainWindow.windowCount,
AppMainWindow.EDITOR_WIDTH,
    AppMainWindow.EDITOR_HEIGHT);
setContentPane(modelingComponent);
setDefaultCloseOperation(DO_NOTHING_ON_CLOSE);
setFrameIcon(null);
setTitle(getTitle() + " " + ++AppMainWindow.windowCount);

// Display some useful information about repaint events
installRepaintListener();
// Installs rubberband selection and handling for some special
// keystrokes such as F2, Control-C, -V, X, A etc.
installHandlers();
installListeners();
}

public void initialize() {
    // Set this internal frame to be selected
    try {
        setSelected(true);
        setMaximum(true);
    } catch (PropertyVetoException e) {
        e.printStackTrace();
    }

    show();
}

public void dispose() {

```

```

boolean canDispose = true;
if (isModified()) {
    switch (JOptionPane.showConfirmDialog(this,
        mxResources.get("saveChanges"))) {
        case JOptionPane.YES_OPTION:
            // TODO salvar as alterações
            // saveChanges();
            break;

        case JOptionPane.CANCEL_OPTION:
            canDispose = false;
    }
}

if (canDispose) {
    List<ModelingEditor> list = mainWindow.getEditors();
    list.remove(this);
    ModelingEditor newEditor = null;
    if (list.size() > 0)
        newEditor = list.get(0); // take first from list

    mainWindow.setCurrentEditor(newEditor);

    super.dispose();
}

protected mxUndoManager createUndoManager() {
    return new mxUndoManager();
}

protected void installHandlers() {
    rubberband = new mxRubberband(modelingComponent);
    rubberband.setBorderColor(Color.BLACK);
    rubberband.setFillColor(new Color(0, 0, 0, 50));
    keyboardHandler = new KeyboardHandler(modelingComponent);
}

public void updateKeyboardHandler() {
    keyboardHandler = new KeyboardHandler(modelingComponent);
}

protected void installRepaintListener() {
    modelingComponent.getGraph().addListener(mxEvent.REPAINT,
        new mxIEventListener() {
            public void invoke(Object source, mxEventObject evt) {
                String buffer = (modelingComponent.getTripleBuffer()
                    : " (unbuffered)");
                mxRectangle dirty = (mxRectangle) evt
                    .getProperty("region");

                if (dirty == null) {
                    mainWindow.status("Repaint all" + buffer,
                        null);
                } else {
                    mainWindow.status("Repaint: x="
                        + (int) (dirty.getX()) + " y="
                        + (int) (dirty.getY()) + " w="
                        + (int) (dirty.getWidth()) + "
                        h="
                        + (int) (dirty.getHeight()) +
                        buffer, null);
                }
            }
        }
    );
}

```

```

        });
    }

    protected void mouseWheelMoved(MouseWheelEvent e) {
        if (e.getWheelRotation() < 0) {
            modelingComponent.zoomIn();
        } else {
            modelingComponent.zoomOut();
        }

        mainWindow.status(mxResources.get("scale")
            + ": "
            + (int) (100 * modelingComponent.getGraph().getView()
                .getScale()) + "%", null);
    }

    protected void showGraphPopupMenu(MouseEvent e) {
        Point pt = SwingUtilities.convertPoint(e.getComponent(), e.getPoint(),
            modelingComponent);
        PopupMenu menu = new PopupMenu(mainWindow);
        menu.show(modelingComponent, pt.x, pt.y);

        e.consume();
    }

    protected void mouseLocationChanged(MouseEvent e) {
        mainWindow.status(e.getX() + ", " + e.getY(), null);
    }

    protected void installListeners() {
        // Installs mouse wheel listener for zooming
        MouseWheelListener wheelTracker = new MouseWheelListener() {
            public void mouseWheelMoved(MouseWheelEvent e) {
                if (e.getSource() instanceof mxGraphOutline
                    || e.isControlDown()) {
                    mouseWheelMoved(e);
                }
            }
        };

        // Handles mouse wheel events in the graph component
        modelingComponent.addMouseListener(wheelTracker);

        // Installs the popup menu in the graph component
        modelingComponent.getGraphControl().addMouseListener(
            new MouseAdapter() {
                public void mouseClicked(MouseEvent e) {
                    modelingManager.insertObjectToModeling(e.getX(),
                        e.getY());
                }

                public void mousePressed(MouseEvent e) {
                    // Handles context menu on the Mac where the trigger
                    // on mousepressed
                    mouseReleased(e);
                }

                public void mouseReleased(MouseEvent e) {
                    if (e.isPopupTrigger()) {
                        showGraphPopupMenu(e);
                    }
                    if (!e.isConsumed() &&

```

is

modelingComponent.isEditEvent(e))

```

        {
            Object cell =
modelingComponent.getCellAt(e.getX(), e.getY(), false);

            if (cell != null &&
modelingComponent.getGraph().isCellEditable(cell) {
                if (((mxCell) cell).getValue()
instanceof ModelingObject)
                    ((ModelingComponent)modelingComponent).startEditingAtCell(cell, e);
            }
        }
    });

    // Installs a mouse motion listener to display the mouse location
    modelingComponent.getGraphControl().addMouseListener(
        new MouseMotionListener() {

            public void mouseDragged(MouseEvent e) {
                mouseLocationChanged(e);
            }

            public void mouseMoved(MouseEvent e) {
                mouseDragged(e);
            }

        });

    addInternalFrameListener(new InternalFrameListener() {

        @Override
        public void internalFrameOpened(InternalFrameEvent arg0) {
        }

        @Override
        public void internalFrameIconified(InternalFrameEvent arg0) {
        }

        @Override
        public void internalFrameDeiconified(InternalFrameEvent arg0) {
        }

        @Override
        public void internalFrameDeactivated(InternalFrameEvent arg0) {
        }

        @Override
        public void internalFrameClosing(InternalFrameEvent arg0) {
            arg0.getInternalFrame().dispose();
        }

        @Override
        public void internalFrameClosed(InternalFrameEvent arg0) {
        }

        @Override
        public void internalFrameActivated(InternalFrameEvent arg0) {
            mainWindow.setCurrentEditor(((ModelingEditor) arg0
                .getInternalFrame());
        }

    });
}

```

```

public void setCurrentFile(File file) {
    File oldValue = currentFile;
    currentFile = file;

    firePropertyChange("currentFile", oldValue, file);

    if (oldValue != file) {
        mainWindow.updateTitle();
    }
}

public File getCurrentFile() {
    return currentFile;
}

/**
 *
 * @param modified
 */
public void setModified(boolean modified) {
    boolean oldValue = this.modified;
    this.modified = modified;

    firePropertyChange("modified", oldValue, modified);

    if (oldValue != modified) {
        mainWindow.updateTitle();
    }
}

/**
 *
 * @return whether or not the current graph has been modified
 */
public boolean isModified() {
    return modified;
}

public boolean isConceptualModeling() {
    return conceptualModeling;
}

public mxGraphComponent getGraphComponent() {
    return modelingComponent;
}

public mxUndoManager getUndoManager() {
    return undoManager;
}

/**
 * Creates an action that executes the specified layout.
 *
 * @param key
 * Key to be used for getting the label from mxResources and also
 * to create the layout instance for the commercial graph editor
 * example.
 * @return an action that executes the specified layout
 */
public Action graphLayout(final String key, boolean animate) {
    final mxIGraphLayout layout = createLayout(key, animate);

    if (layout != null) {
        return new AbstractAction(mxResources.get(key)) {

```

```

        public void actionPerformed(ActionEvent e) {
            final mxGraph graph = modelingComponent.getGraph();
            Object cell = graph.getSelectionCell();

            if (cell == null
                || graph.getModel().getChildCount(cell) == 0)
            {
                cell = graph.getDefaultParent();
            }

            graph.getModel().beginUpdate();
            try {
                long t0 = System.currentTimeMillis();
                layout.execute(cell);
                mainWindow.status("Layout: "
                    + (System.currentTimeMillis() - t0) +
" ms", null);
            } finally {
                mxMorphing morph = new
                    (20, 1.2, 20);

                morph.addListener(mxEvent.DONE, new
                    mxEventListener() {

                        public void invoke(Object sender,
                            mxEventObject evt) {

                                graph.getModel().endUpdate();
                            }

                        });

                morph.startAnimation();
            }
        }
    } else {
        return new AbstractAction(mxResources.get(key)) {

            public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(modelingComponent,
                    mxResources.get("noLayout"));
            }

        };
    }
}

/**
 * Creates a layout instance for the given identifier.
 */
protected mxIGraphLayout createLayout(String ident, boolean animate) {
    mxIGraphLayout layout = null;

    if (ident != null) {
        mxGraph graph = modelingComponent.getGraph();

        if (ident.equals("verticalHierarchical")) {
            layout = new mxHierarchicalLayout(graph);
        } else if (ident.equals("horizontalHierarchical")) {
            layout = new mxHierarchicalLayout(graph, JLabel.WEST);
        } else if (ident.equals("verticalTree")) {
            layout = new mxCompactTreeLayout(graph, false);
        }
    }
}

```



```

    } else if (ident.equals("horizontalTree")) {
        layout = new mxCompactTreeLayout(graph, true);
    } else if (ident.equals("parallelEdges")) {
        layout = new mxParallelEdgeLayout(graph);
    } else if (ident.equals("placeEdgeLabels")) {
        layout = new mxEdgeLabelLayout(graph);
    } else if (ident.equals("organicLayout")) {
        layout = new mxOrganicLayout(graph);
    }
    if (ident.equals("verticalPartition")) {
        layout = new mxPartitionLayout(graph, false) {
            /**
             * Overrides the empty implementation to return the size of
             * the graph control.
             */
            public mxRectangle getContainerSize() {
                return modelingComponent.getLayoutAreaSize();
            }
        };
    } else if (ident.equals("horizontalPartition")) {
        layout = new mxPartitionLayout(graph, true) {
            /**
             * Overrides the empty implementation to return the size of
             * the graph control.
             */
            public mxRectangle getContainerSize() {
                return modelingComponent.getLayoutAreaSize();
            }
        };
    } else if (ident.equals("verticalStack")) {
        layout = new mxStackLayout(graph, false) {
            /**
             * Overrides the empty implementation to return the size of
             * the graph control.
             */
            public mxRectangle getContainerSize() {
                return modelingComponent.getLayoutAreaSize();
            }
        };
    } else if (ident.equals("horizontalStack")) {
        layout = new mxStackLayout(graph, true) {
            /**
             * Overrides the empty implementation to return the size of
             * the graph control.
             */
            public mxRectangle getContainerSize() {
                return modelingComponent.getLayoutAreaSize();
            }
        };
    } else if (ident.equals("circleLayout")) {
        layout = new mxCircleLayout(graph);
    }
}

return layout;
}
}

```

```
package ufsc.sisinf.brmodelo2all.control;
```

```
import java.util.ArrayList;
import java.util.List;
```

```

import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.SwingUtilities;

import ufsc.sisinf.brmodelo2all.model.Cardinality;
import ufsc.sisinf.brmodelo2all.model.ModelingComponent;
import ufsc.sisinf.brmodelo2all.model.objects.AssociativeEntityObject;
import ufsc.sisinf.brmodelo2all.model.objects.AssociativeRelationObject;
import ufsc.sisinf.brmodelo2all.model.objects.AttributeObject;
import ufsc.sisinf.brmodelo2all.model.objects.ColumnObject;
import ufsc.sisinf.brmodelo2all.model.objects.ConnectorObject;
import ufsc.sisinf.brmodelo2all.model.objects.EntityObject;
import ufsc.sisinf.brmodelo2all.model.objects.InheritanceObject;
import ufsc.sisinf.brmodelo2all.model.objects.ModelingObject;
import ufsc.sisinf.brmodelo2all.model.objects.RelationObject;
import ufsc.sisinf.brmodelo2all.model.objects.TableObject;
import ufsc.sisinf.brmodelo2all.ui.AppMainWindow;
import ufsc.sisinf.brmodelo2all.util.Messages;

import com.mxgraph.model.mxCell;
import com.mxgraph.model.mxGeometry;
import com.mxgraph.model.mxICell;
import com.mxgraph.model.mxIGraphModel;
import com.mxgraph.swing.mxGraphComponent;
import com.mxgraph.util.mxConstants;
import com.mxgraph.util.mxPoint;
import com.mxgraph.util.mxRectangle;
import com.mxgraph.util.mxResources;
import com.mxgraph.view.mxCellState;
import com.mxgraph.view.mxGraph;

public class ModelingManager {

    protected final AppMainWindow mainWindow;
    protected ModelingComponent modelingComponent;

    protected Object source = null;
    protected Object target = null;

    protected enum Connection {
        //TODO melhorar o enum TABLES
        ENTITY_TO_ENTITY, ENTITY_TO_RELATION, RELATION_TO_ENTITY,
ENTITY_TO_INHERITANCE,
        TABLES, NONE, ERROR
    }

    public ModelingManager(final AppMainWindow mainWindow,
        mxGraphComponent modelingComponent) {
        this.mainWindow = mainWindow;
        this.modelingComponent = (ModelingComponent)modelingComponent;
    }

    public void insertObjectToModeling(double x, double y) {
        String selectedEntry = mainWindow.getSelectedEntry();

        if (selectedEntry != null) {
            boolean clearSelection = false;
            mxGraph graph = modelingComponent.getGraph();
            Object insertedCell = null;

            if (selectedEntry == mxResources.get("entity")) //$NON-NLS-1$
                insertedCell = graph.insertVertex(graph.getDefaultParent(), null,
                    new EntityObject(mxResources.get("entity"), x, y,
120, 50, //$NON-NLS-1$
                        "entity")); //$NON-NLS-1$

```

```

else if (selectedEntry == mxResources.get("relation")) //$NON-NLS-1$
    insertedCell = graph.insertVertex(graph.getDefaultParent(), null,
        new RelationObject(mxResources.get("relation")), x, y,
//$NON-NLS-1$
        120, 50, "relation"); //$NON-NLS-1$
else if (selectedEntry == mxResources.get("selfRelation")) //$NON-NLS-1$
    insertedCell = insertSelfRelatedObject(x, y);
else if (selectedEntry == mxResources.get("inheritance")) //$NON-NLS-1$
    insertedCell = insertInheritanceObject(x, y);
else if (selectedEntry == mxResources.get("exclusiveInheritance")) //$NON-NLS-1$
    insertedCell = insertExclusiveInheritanceObject(x, y);
else if (selectedEntry == mxResources.get("nonExclusiveInheritance")) //$NON-NLS-1$
    insertedCell = insertNonExclusiveInheritanceObject(x, y);
else if (selectedEntry == mxResources.get("associativeEntity")) //$NON-NLS-1$
    insertedCell = insertAssociativeEntityObject(x, y);
else if (selectedEntry == mxResources.get("composedAttribute")) //$NON-NLS-1$
    insertedCell = insertComposedAttributeObject(x, y);
else if (selectedEntry == mxResources.get("attribute")) //$NON-NLS-1$
    || selectedEntry == mxResources.get("identifierAttribute"))
//$NON-NLS-1$
    insertedCell = insertAttributeObject((int) x, (int) y);
else if (selectedEntry == mxResources.get("connector")) //$NON-NLS-1$
    clearSelection = insertConnectorObject((int) x, (int) y);
else if (selectedEntry == mxResources.get("table")) //$NON-NLS-1$
    insertedCell = graph.insertVertex(graph.getDefaultParent(), null,
        new TableObject(mxResources.get("table")), x, y, 150,
30, //$NON-NLS-1$
        "table"); //$NON-NLS-1$
else if (selectedEntry == mxResources.get("primaryKey")) //$NON-NLS-1$
    || selectedEntry == mxResources.get("foreignKey")) //$NON-NLS-1$
    || selectedEntry == mxResources.get("field")) //$NON-NLS-1$
    insertedCell = insertColumnObject(x, y, selectedEntry);

// SETAR AQUI PARA DESELECIONAR O BOTÃO
if (insertedCell != null) {
    // clear selection
    mainWindow.clearModelingPalette();
    mainWindow.status(null, null);
    source = null;
    target = null;

    mainWindow.properties(insertedCell, modelingComponent);
} else if (clearSelection) {
    mainWindow.clearModelingPalette();
    mainWindow.status(null, null);
    source = null;
    target = null;
}
}

public Object insertColumnObject(double x, double y, String columnType) {
    Object columnObject = null;
    Object object = modelingComponent.getCellAt((int) x, (int) y);
    String style = "column"; //$NON-NLS-1$
    if (columnType == mxResources.get("primaryKey")) //$NON-NLS-1$
        style = "primaryKey"; //$NON-NLS-1$
    else if (columnType == mxResources.get("foreignKey")) //$NON-NLS-1$
        style = "foreignKey"; //$NON-NLS-1$
}

```

```

    if (object instanceof mxICell) {
        mxGraph graph = modelingComponent.getGraph();
        mxICell sourceCell = (mxICell) object;

        if (sourceCell.getStyle() == "table") { //$NON-NLS-1$
            mxGeometry geometry = sourceCell.getGeometry();
            double width = geometry.getWidth();
            double height = 20;
            double newY = (sourceCell.getChildCount() * 20) + 25;
            mxRectangle newBounds = new
mxRectangle(geometry.getRectangle());
            newBounds.setHeight(newBounds.getHeight() + 25);
            graph.resizeCell(sourceCell, newBounds);
            columnObject = graph.insertVertex(sourceCell, null,
                new ColumnObject(columnType, sourceCell, style ==
"primaryKey", style == "foreignKey"), 0, newY, width, height,
                style);
            sourceCell.insert((mxICell) columnObject);
            ((TableObject)sourceCell.getValue()).addChildObject(columnObject);

        } else if ((sourceCell.getStyle() == "column" || sourceCell.getStyle() ==
"primaryKey" ||
== "bothKeys") //$NON-NLS-1$
                sourceCell.getStyle() == "foreignKey" || sourceCell.getStyle()
                && sourceCell.getParent().getStyle() == "table") { //$NON-
NLS-1$
            mxICell tableCell = sourceCell.getParent();
            mxGeometry geometry = tableCell.getGeometry();
            double width = geometry.getWidth();
            double height = 20;
            double newY = (tableCell.getChildCount() * 20) + 25;
            mxRectangle newBounds = new
mxRectangle(geometry.getRectangle());
            newBounds.setHeight(newBounds.getHeight() + 25);
            graph.resizeCell(tableCell, newBounds);
            columnObject = graph.insertVertex(tableCell, null,
                new ColumnObject(columnType, tableCell, style ==
"primaryKey", style == "foreignKey"), 0, newY, width, height,
                style);
            tableCell.insert((mxICell) columnObject);
            ((TableObject)tableCell.getValue()).addChildObject(columnObject);

        } else
            JOptionPane.showMessageDialog((JFrame)
SwingUtilities.getWindowForComponent(mainWindow),

                Messages.getString("ModelingManager.warning.notAnTable"),

                Messages.getString("ModelingManager.warning.window.advice"), 0); //$NON-NLS-1$ //$NON-
NLS-2$

        } else
            JOptionPane.showMessageDialog((JFrame)
SwingUtilities.getWindowForComponent(mainWindow),

                Messages.getString("ModelingManager.warning.needs.selectTableToInsertField"),

                Messages.getString("ModelingManager.warning.window.advice"), 0); //$NON-NLS-1$ //$NON-
NLS-2$

        return columnObject;
    }

    public Object insertInheritanceObject(double x, double y) {
        Object inheritanceObject = null;
        Object object = modelingComponent.getCellAt((int) x, (int) y);

```

```

        if (object instanceof mxICell) {
            mxGraph graph = modelingComponent.getGraph();
            mxICell sourceCell = (mxICell) object;

            if (sourceCell.getStyle() == "entity") { //$NON-NLS-1$
                graph.getModel().beginUpdate();
                try {
                    mxGeometry geometry = sourceCell.getGeometry();
                    double newX = geometry.getX() + (geometry.getWidth() / 2) -
15;
                    double newY = geometry.getY() + (geometry.getHeight() *
1.5);

                    EntityObject entity = (EntityObject) sourceCell.getValue();

                    inheritanceObject = graph.insertVertex(
                        graph.getDefaultParent(), null, new
InheritanceObject(mxResources.get("inheritance"), sourceCell), newX, //$NON-NLS-1$
                        newY, 30, 30, "inheritance"); //$NON-NLS-
1$

                    graph.insertEdge(object, "straight", null, object, //$NON-NLS-
1$
                        inheritanceObject, "straight"); //$NON-NLS-
1$

                    entity.setSpecialized(true);
                    entity.addChildObject(inheritanceObject);
                } finally {
                    graph.getModel().endUpdate();
                }
            } else {
                JFrame frame = (JFrame)
SwingUtilities.windowForComponent(mainWindow);
                JOptionPane.showMessageDialog(frame,
Messages.getString("ModelingManager.warning.notAnEntity"),
Messages.getString("ModelingManager.warning.window.advice"), 0); //$NON-NLS-1$ //$NON-NLS-2$
            }
            } else {
                JFrame frame = (JFrame) SwingUtilities.windowForComponent(mainWindow);
                JOptionPane.showMessageDialog(frame,
Messages.getString("ModelingManager.warning.needs.selectEntityToSpecialize"), "Aviso", 0); //$NON-
NLS-1$
            }

            return inheritanceObject;
        }

        public Object insertExclusiveInheritanceObject(double x, double y) {
            Object inheritanceObject = insertInheritanceObject(x, y);
            if (inheritanceObject != null) {
                mxGraph graph = modelingComponent.getGraph();
                graph.getModel().beginUpdate();
                try {
                    mxICell sourceCell = (mxICell) inheritanceObject;
                    mxGeometry geometry = sourceCell.getGeometry();
                    double centerX = geometry.getCenterX();
                    double newY = geometry.getY() + (geometry.getHeight() * 1.5);
                    double newX = centerX - 150;

                    InheritanceObject inheritance =
(InheritanceObject)((mxCell)inheritanceObject).getValue();
                    inheritance.setExclusive(true);
                    EntityObject entity1 = new EntityObject(mxResources.get("entity"));
                }
            }
        }
    }
}
//$NON-NLS-1$

```

```

entity1.setParentObject(inheritanceObject);
Object newObject = graph.insertVertex(graph.getDefaultParent(),
    null, entity1, newX, newY,
    120, 50, "entity"); //$NON-NLS-1$
graph.insertEdge(graph.getDefaultParent(), "straight", null, //$NON-NLS-1$
    newObject, inheritanceObject, "straight"); //$NON-NLS-1$
inheritance.addChildObject(newObject);

newX = centerX + 30;
EntityObject entity2 = new EntityObject(mxResources.get("entity"));
//$NON-NLS-1$
entity2.setParentObject(inheritanceObject);
newObject = graph.insertVertex(graph.getDefaultParent(), null,
    entity2, newX, newY, 120,
    50, "entity"); //$NON-NLS-1$
graph.insertEdge(graph.getDefaultParent(), "straight", null, //$NON-NLS-1$
    newObject, inheritanceObject, "straight"); //$NON-NLS-1$
inheritance.addChildObject(newObject);

    } finally {
        graph.getModel().endUpdate();
    }
}

return inheritanceObject;
}

public Object insertNonExclusiveInheritanceObject(double x, double y) {
    Object inheritanceObject = insertInheritanceObject(x, y);
    if (inheritanceObject != null) {
        mxGraph graph = modelingComponent.getGraph();
        graph.getModel().beginUpdate();
        try {
            mxICell sourceCell = (mxICell) inheritanceObject;
            mxGeometry geometry = sourceCell.getGeometry();
            double centerX = geometry.getCenterX();
            double newY = geometry.getY() + (geometry.getHeight() * 1.5);
            double newX = centerX - 60;
            InheritanceObject inheritance =
(InheritanceObject)sourceCell.getValue();
            EntityObject entity = new EntityObject(mxResources.get("entity"));
            //$NON-NLS-1$
            entity.setParentObject(inheritanceObject);
            Object newObject = graph.insertVertex(graph.getDefaultParent(),
                null, entity, newX, newY,
                120, 50, "entity"); //$NON-NLS-1$
            graph.insertEdge(graph.getDefaultParent(), "straight", null, //$NON-NLS-1$
                newObject, inheritanceObject, "straight"); //$NON-NLS-1$
            inheritance.addChildObject(newObject);
        } finally {
            graph.getModel().endUpdate();
        }
    }

    return inheritanceObject;
}

public Object insertAssociativeEntityObject(double x, double y) {
    Object associativeEntityObject = null;

```

```

        mxGraph graph = modelingComponent.getGraph();
        graph.getModel().beginUpdate();
        try {
            AssociativeRelationObject relationObject = new
AssociativeRelationObject(mxResources.get("relation")); //$NON-NLS-1$
            AssociativeEntityObject entityObject = new
AssociativeEntityObject(mxResources.get("associativeEntity"), relationObject); //$NON-NLS-1$
            associativeEntityObject = graph.insertVertex(graph.getDefaultParent(), null,
entityObject, x, y, 150, 80, "associativeEntity"); //$NON-NLS-
1$

            entityObject.addChildObject(graph.insertVertex(associativeEntityObject, null,
relationObject, 15, 20, 120, 50, "associativeRelation")); //$NON-NLS-1$
        } finally {
            graph.getModel().endUpdate();
        }

        return associativeEntityObject;
    }

    public Object insertComposedAttributeObject(double x, double y) {
        Object attributeObject = insertAttributeObject((int) x, (int) y);
        if (attributeObject != null) {
            mxGraph graph = modelingComponent.getGraph();
            graph.getModel().beginUpdate();
            try {
                mxICell attributeCell = (mxICell) attributeObject;
                AttributeObject attribute = (AttributeObject)attributeCell.getValue();
                attribute.setComposed(true);

                mxGeometry geometry = attributeCell.getGeometry();
                double newY = AdjustAttributePosition(geometry.getY());
                double newX = geometry.getCenterX() - 40;
                Object[] atributo = new Object[1];

                atributo[0] = graph.insertVertex(graph.getDefaultParent(), null,
new AttributeObject(mxResources.get("attribute"),
attributeObject, false), newX, newY, 10, 10, //$NON-NLS-1$
"attribute"); //$NON-NLS-1$

                attribute.addChildObject(atributo[0]);

                graph.setCellStyles(mxConstants.STYLE_LABEL_POSITION,
mxConstants.ALIGN_LEFT, atributo);
                graph.setCellStyles(mxConstants.STYLE_ALIGN,
mxConstants.ALIGN_RIGHT, atributo);

                graph.insertEdge(graph.getDefaultParent(), "straight", null, //$NON-NLS-1$
attributeObject, atributo[0], "straight"); //$NON-NLS-1$
1$

                newX = geometry.getCenterX() + 40;
                atributo[0] = graph.insertVertex(graph.getDefaultParent(), null,
new AttributeObject(mxResources.get("attribute"),
attributeObject, false), newX, newY, 10, 10, //$NON-NLS-1$
"attribute"); //$NON-NLS-1$

                attribute.addChildObject(atributo[0]);

                graph.setCellStyles(mxConstants.STYLE_LABEL_POSITION,
mxConstants.ALIGN_RIGHT, atributo);
                graph.setCellStyles(mxConstants.STYLE_ALIGN,
mxConstants.ALIGN_LEFT, atributo);

```

```

graph.insertEdge(graph.getDefaultParent(), "straight", null, //$NON-
NLS-1$
attributeObject, atributo[0], "straight");
//$NON-NLS-1$
} finally {
graph.getModel().endUpdate();
}
}
return attributeObject;
}

public Object insertAttributeObject(int x, int y) {
Object attributeObject = null;
Object object = modelingComponent.getCellAt(x, y);

if (object instanceof mxICell) {
mxGraph graph = modelingComponent.getGraph();
mxICell sourceCell = (mxICell) object;

if (sourceCell.isVertex()) {
if (!(sourceCell.getValue() instanceof InheritanceObject)) {

graph.getModel().beginUpdate();
try {
String selectedEntry =
mainWindow.getSelectedEntry();

double originalY;
if (sourceCell.getParent().getValue() instanceof
ModelingObject)
originalY =
sourceCell.getParent().getGeometry().getY();
else
originalY = sourceCell.getGeometry().getY();

if (sourceCell.getValue() instanceof AttributeObject) {
AttributeObject attribute =
(AttributeObject)sourceCell.getValue();
attribute.setComposed(true);
}

double newY = AdjustAttributePosition(originalY);

boolean identifier = selectedEntry ==
mxResources.get("identifierAttribute") ? true : false; //$NON-NLS-1$
String style = identifier ? "identifierAttribute" :
"attribute"; //$NON-NLS-1$ //$NON-NLS-2$
attributeObject =
graph.insertVertex(graph.getDefaultParent(),
null, new
AttributeObject(selectedEntry, sourceCell, identifier), x, newY, 10, 10, style);

graph.insertEdge(graph.getDefaultParent(), "straight",
null, //$NON-NLS-1$
object, attributeObject, "straight");
//$NON-NLS-1$

((ModelingObject)
sourceCell.getValue()).addChildObject(attributeObject);
} finally {
graph.getModel().endUpdate();
}
} else {
JFrame frame = (JFrame)
SwingUtilities.getWindowForComponent(mainWindow);

```



```

        JOptionPane.showMessageDialog(frame,
Messages.getString("ModelingManager.warning.cantInsertAttributeOnSpecialization"), "Aviso", 0);
        //NON-NLS-1$
    }
    }
    } else {
        JFrame frame = (JFrame) SwingUtilities.getWindowForComponent(mainWindow);
        JOptionPane.showMessageDialog(frame,
Messages.getString("ModelingManager.warnig.needs.selectAnEntityRelationshipAssociativeAttribute"),
"Aviso", 0); //NON-NLS-1$
    }

    return attributeObject;
}

public double AdjustAttributePosition(double cellY) {
    return cellY - 30;
}

public Object insertSelfRelatedObject(double x, double y) {
    Object returnObject = null;
    Object object = modelingComponent.getCellAt((int) x, (int) y);

    if (object instanceof mxICell) {
        mxGraph graph = modelingComponent.getGraph();
        mxICell sourceCell = (mxICell) object;

        if (sourceCell.getStyle() == "associativeRelation") { //NON-NLS-1$
            sourceCell = sourceCell.getParent();
        }

        if (sourceCell.getStyle() == "entity" || sourceCell.getStyle() ==
"associativeEntity") { //NON-NLS-1$ //NON-NLS-2$
            graph.getModel().beginUpdate();
            Object[] objects = new Object[5];
            try {
                mxGeometry geometry = sourceCell.getGeometry();
                double newX = geometry.getX() + (geometry.getWidth() * 2);
                double newY = geometry.getY();

                objects[0] = sourceCell;
                objects[4] = sourceCell;
                RelationObject relationObject = new
RelationObject(mxResources.get("relation"), sourceCell); //NON-NLS-1$

                objects[2] = graph.insertVertex(graph.getDefaultParent(), null,
relationObject, newX,
newY, 120, 50, "relation"); //NON-NLS-1$
                objects[1] = graph.insertEdge(graph.getDefaultParent(), null,
new
ConnectorObject(Cardinality.getValue("(0,n)"), //NON-NLS-1$
sourceCell, objects[2],
"entityRelationConnector"); //NON-NLS-1$
                objects[3] = graph.insertEdge(graph.getDefaultParent(), null,
new
ConnectorObject(Cardinality.getValue("(0,n)"), //NON-NLS-1$
objects[2], sourceCell,
"entityRelationConnector"); //NON-NLS-1$

                if (sourceCell.getValue() instanceof EntityObject) {
                    EntityObject entity = (EntityObject)
sourceCell.getValue();

                    entity.setSelfRelated(true);
                    entity.addChildObject(objects[2]);
                    relationObject.addRelatedObject(entity);
                }
            } catch (Exception e) {
                Messages.showErrorDialog(e.getMessage(), "Error");
            }
        }
    }
}

```

```

        relationObject.addRelatedObject(entity);
        entity.addRelation(relationObject);
    } else if (sourceCell.getValue() instanceof
AssociativeEntityObject) {
        AssociativeEntityObject entity =
(AssociativeEntityObject) sourceCell.getValue();
        entity.setSelfRelated(true);
        entity.addChildObject(objects[2]);
        relationObject.addRelatedObject(entity);
        relationObject.addRelatedObject(entity);
        entity.addRelation(relationObject);
    }

    returnObject = objects[2];
} finally {
    graph.getModel().endUpdate();
}

AdjustEdgeLabelPosition(objects);
} else {
    JFrame frame = (JFrame)
SwingUtilities.windowForComponent(mainWindow);
    JOptionPane.showMessageDialog(frame,
Messages.getString("ModelingManager.warning.needs.selectAnEntityAssociative"), "Aviso", 0); //$NON-
NLS-1$
    }
} else {
    JFrame frame = (JFrame) SwingUtilities.windowForComponent(mainWindow);
    JOptionPane.showMessageDialog(frame,
Messages.getString("ModelingManager.warning.needs.selectAnEntityAssociative"), "Aviso", 0); //$NON-
NLS-1$
    }

    return returnObject;
}

public boolean insertConnectorObject(int x, int y) {
    boolean clearSelection = false;

    if (source == null) {
        source = modelingComponent.getCellAt((int) x, (int) y);

        if (source instanceof mxICell) {
            mxICell sourceCell = (mxICell) source;

            if (!sourceCell.isVertex() || sourceCell.getValue() instanceof
AttributeObject) {
                source = null;
                JOptionPane.showMessageDialog((JFrame)
SwingUtilities.windowForComponent(mainWindow),
"Este não é um objeto válido para realizar uma
conexão", "Aviso", 0); //$NON-NLS-1$
            }

            } else
                JOptionPane.showMessageDialog((JFrame)
SwingUtilities.windowForComponent(mainWindow),
"É necessário selecionar um objeto", "Aviso", 0);
//$NON-NLS-1$

        } else if (target == null) {
            target = modelingComponent.getCellAt((int) x, (int) y);

            if (target instanceof mxICell) {

```

```

        mxICell targetCell = (mxICell) target;

        if (!targetCell.isVertex() || targetCell.getValue() instanceof
AttributeObject) {
            target = null;
            JOptionPane.showMessageDialog((JFrame)
SwingUtilities.getWindowForComponent(mainWindow),
            "Este não é um objeto válido para realizar uma
conexão", "Aviso", 0); //NON-NLS-1$
        } else{
            insertConnectorBetweenObjects();
            clearSelection = true;
        }
    } else
        JOptionPane.showMessageDialog((JFrame)
SwingUtilities.getWindowForComponent(mainWindow),
            "É necessário selecionar um objeto", "Aviso", 0);
//NON-NLS-1$
    }

    return clearSelection;
}

public Connection getConnection(String sourceStyle, String targetStyle) {
    Connection connection = Connection.NONE;

    if ((sourceStyle == "entity" || sourceStyle == "associativeEntity") && //NON-NLS-1$
//NON-NLS-2$
        (targetStyle == "entity" || targetStyle == "associativeEntity") ) //NON-NLS-1$
//NON-NLS-2$
        connection = Connection.ENTITY_TO_ENTITY;
    else if ((sourceStyle == "entity" || sourceStyle == "associativeEntity") && //NON-NLS-1$
//NON-NLS-2$
        (targetStyle == "relation" || targetStyle == "associativeRelation"))
//NON-NLS-1$ //NON-NLS-2$
        connection = Connection.ENTITY_TO_RELATION;
    else if ((sourceStyle == "relation" || sourceStyle == "associativeRelation") && //NON-
NLS-1$ //NON-NLS-2$
        (targetStyle == "entity" || targetStyle == "associativeEntity")) //NON-
NLS-1$ //NON-NLS-2$
        connection = Connection.RELATION_TO_ENTITY;
    else if ((sourceStyle == "table" || sourceStyle == "column") && //NON-NLS-1$
//NON-NLS-2$
        (targetStyle == "table" || targetStyle == "column")) //NON-NLS-1$
//NON-NLS-2$
        connection = Connection.TABLES;
    else if ((sourceStyle == "inheritance" && targetStyle == "entity") || //NON-NLS-1$
//NON-NLS-2$
        (sourceStyle == "entity" && targetStyle == "inheritance")) //NON-
NLS-1$ //NON-NLS-2$
        connection = Connection.ENTITY_TO_INHERITANCE;
    else if ((sourceStyle == "relation" && targetStyle == "relation") ||
        (sourceStyle == "inheritance" && targetStyle == "inheritance") ||
        (sourceStyle == "relation" && targetStyle == "inheritance") ||
        (sourceStyle == "inheritance" && targetStyle == "relation"))
        connection = Connection.ERROR;

    return connection;
}

public void insertConnectorBetweenObjects() {
    mxGraph graph = modelingComponent.getGraph();
    mxICell sourceCell = (mxICell) source;
    mxICell targetCell = (mxICell) target;
    Object[] objects = null;

```

```

Connection connection = getConnection(sourceCell.getStyle(), targetCell.getStyle());

switch(connection) {
case ENTITY_TO_ENTITY:
    mxGeometry sourceGeometry = sourceCell.getGeometry();
    if (source == target) {
        Object object =
modelingComponent.getCellAt((int)sourceGeometry.getCenterX(), (int)sourceGeometry.getCenterY());

        if (object instanceof mxICell) {
            if (sourceCell.getStyle() == "associativeRelation") //$NON-
NLS-1$
                sourceCell = sourceCell.getParent();

                if (sourceCell.getStyle() == "entity" || sourceCell.getStyle() ==
"associativeEntity") { //$NON-NLS-1$ //$NON-NLS-2$
                    graph.getModel().beginUpdate();
                    objects = new Object[5];
                    try {
                        mxGeometry geometry =
sourceCell.getGeometry();
                        double newX = geometry.getX() +
(geometry.getWidth() * 2);
                        double newY = geometry.getY();

                        objects[0] = sourceCell;
                        objects[4] = sourceCell;
                        RelationObject relationObject = new
RelationObject(mxResources.get("relation"), sourceCell); //$NON-NLS-1$
                        objects[2] =
graph.insertVertex(graph.getDefaultParent(), null,
relationObject, newX,
newY, 120, 50, "relation");
                        //$NON-NLS-1$
                        objects[1] =
graph.insertEdge(graph.getDefaultParent(), null,
new
ConnectorObject(Cardinality.getValue("(0,n)"), //$NON-NLS-1$
"entityRelationConnector"); //$NON-NLS-1$
                        objects[3] =
graph.insertEdge(graph.getDefaultParent(), null,
new
ConnectorObject(Cardinality.getValue("(0,n)"), //$NON-NLS-1$
"entityRelationConnector"); //$NON-NLS-1$
                        EntityObject) {
                            EntityObject entity = (EntityObject)
sourceCell.getValue();
                            entity.setSelfRelated(true);
                            entity.addChildObject(objects[2]);

                            relationObject.addRelatedObject(entity);
                            relationObject.addRelatedObject(entity);
                            entity.addRelation(relationObject);
                        } else if (sourceCell.getValue() instanceof
AssociativeEntityObject) {
                            AssociativeEntityObject entity =
(AssociativeEntityObject) sourceCell.getValue();

```

```

entity.setSelfRelated(true);
entity.addChildObject(objects[2]);

relationObject.addRelatedObject(entity);

relationObject.addRelatedObject(entity);

entity.addRelation(relationObject);
}
} finally {
graph.getModel().endUpdate();
}
AdjustEdgeLabelPosition(objects);
} else {
JFrame frame = (JFrame)
SwingUtilities.windowForComponent(mainWindow);
OptionPane.showMessageDialog(frame,
Messages.getString("ModelingManager.warning.needs.selectAnEntityAssociative"), "Aviso", 0); //$NON-NLS-1$
}
} else {
JFrame frame = (JFrame)
SwingUtilities.windowForComponent(mainWindow);
OptionPane.showMessageDialog(frame,
Messages.getString("ModelingManager.warning.needs.selectAnEntityAssociative"), "Aviso", 0); //$NON-NLS-1$
}
} else {
mxGeometry targetGeometry = targetCell.getGeometry();
double srcX = sourceGeometry.getX();
double srcY = sourceGeometry.getY();
double trgtX = targetGeometry.getX();
double trgtY = targetGeometry.getY();
double x = srcX > trgtX ? trgtX + ((srcX - trgtX) / 2) : srcX + ((trgtX -
srcX) / 2);
double y = srcY > trgtY ? trgtY + ((srcY - trgtY) / 2) : srcY + ((trgtY -
srcY) / 2);
if (x == srcX && x == trgtX && y == srcY && y == trgtY)
y += sourceGeometry.getHeight() +
(sourceGeometry.getHeight() * 1.5);

objects = new Object[5];
objects[0] = source;
objects[4] = target;
RelationObject relationObject = new
RelationObject(mxResources.get("relation")); //$NON-NLS-1$

relationObject.addRelatedObject((ModelingObject)((mxCell)source).getValue());

((ModelingObject)((mxCell)source).getValue()).addRelation(relationObject);

relationObject.addRelatedObject((ModelingObject)((mxCell)target).getValue());

((ModelingObject)((mxCell)target).getValue()).addRelation(relationObject);
objects[2] = graph.insertVertex(graph.getDefaultParent(), null,
relationObject, x, y, 120, 50, "relation"); //$NON-NLS-1$
ConnectorObject firstConnector = new
ConnectorObject(Cardinality.getValue("(0,n)")); //$NON-NLS-1$
objects[1] = graph.insertEdge(graph.getDefaultParent(), null,
firstConnector, source, objects[2], "entityRelationConnector"); //$NON-NLS-1$
ConnectorObject secondConnector = new
ConnectorObject(Cardinality.getValue("(0,n)")); //$NON-NLS-1$

```

```

        objects[3] = graph.insertEdge(graph.getDefaultParent(), null,
secondConnector, objects[2], target, "entityRelationConnector"); //$NON-NLS-1$
    }
    break;

    case ENTITY_TO_RELATION:
        objects = new Object[3];
        objects[0] = source;
        objects[2] = target;
        if (((mxCell)source).getValue() instanceof RelationObject){
            RelationObject relationObject =
(RelationObject)((mxCell)source).getValue();
            ModelingObject relatedObject =
(ModelingObject)((mxCell)target).getValue();
            relationObject.addRelatedObject(relatedObject);
            relatedObject.addRelation(relationObject);

        } else {
            RelationObject relationObject =
(RelationObject)((mxCell)target).getValue();
            ModelingObject relatedObject =
(ModelingObject)((mxCell)source).getValue();
            relationObject.addRelatedObject(relatedObject);
            relatedObject.addRelation(relationObject);

        }

        ConnectorObject connector = new
ConnectorObject(Cardinality.getValue("(0,n)")); //$NON-NLS-1$
        objects[1] = graph.insertEdge(graph.getDefaultParent(), null, connector, source,
target, "entityRelationConnector"); //$NON-NLS-1$
        break;

    case RELATION_TO_ENTITY:
        objects = new Object[3];
        objects[0] = target;
        objects[2] = source;
        if (((mxCell)source).getValue() instanceof RelationObject){
            RelationObject relationObject =
(RelationObject)((mxCell)source).getValue();
            ModelingObject relatedObject =
(ModelingObject)((mxCell)target).getValue();
            relationObject.addRelatedObject(relatedObject);
            relatedObject.addRelation(relationObject);

        } else {
            RelationObject relationObject =
(RelationObject)((mxCell)target).getValue();
            ModelingObject relatedObject =
(ModelingObject)((mxCell)source).getValue();
            relationObject.addRelatedObject(relatedObject);
            relatedObject.addRelation(relationObject);

        }

        ConnectorObject connector2 = new
ConnectorObject(Cardinality.getValue("(0,n)")); //$NON-NLS-1$
        objects[1] = graph.insertEdge(graph.getDefaultParent(), null, connector2, target,
source, "entityRelationConnector"); //$NON-NLS-1$
        break;

    case ENTITY_TO_INHERITANCE:
        InheritanceObject inheritance = null;
        EntityObject entity = null;
        Object child = null;

```

```

        if (sourceCell.getValue() instanceof InheritanceObject) {
            inheritance = (InheritanceObject)sourceCell.getValue();
            entity = (EntityObject)targetCell.getValue();
            child = targetCell;
        } else {
            inheritance = (InheritanceObject)targetCell.getValue();
            entity = (EntityObject)sourceCell.getValue();
            child = sourceCell;
        }

        inheritance.addChildObject(child);
        entity.setParentObject(child == sourceCell ? targetCell : sourceCell);
        if (inheritance.getChildObjects().size() == 2)
            inheritance.setExclusive(true);
        graph.insertEdge(graph.getDefaultParent(), "straight", null, source, target,
"straight"); //$NON-NLS-1$ //$NON-NLS-2$
        break;

    case TABLES:
        sourceGeometry = sourceCell.getGeometry();
        mxGeometry targetGeometry = targetCell.getGeometry();
        double srcX = sourceGeometry.getX();
        double srcY = sourceGeometry.getY();
        double trgtX = targetGeometry.getX();
        double trgtY = targetGeometry.getY();
        double x = srcX > trgtX ? trgtX + ((srcX - trgtX) / 2) : srcX + ((trgtX - srcX) / 2);
        double y = srcY > trgtY ? trgtY + ((srcY - trgtY) / 2) : srcY + ((trgtY - srcY) / 2);

        if (x == srcX && x == trgtX && y == srcY && y == trgtY)
            y += sourceGeometry.getHeight() + (sourceGeometry.getHeight() * 1.5);

        objects = new Object[5];
        objects[0] = sourceCell.getStyle() == "table" ? source : sourceCell.getParent();
        //$NON-NLS-1$
        objects[4] = targetCell.getStyle() == "table" ? target : targetCell.getParent();
        //$NON-NLS-1$
        objects[2] = graph.insertVertex(graph.getDefaultParent(), null, "", x, y, 20, 10,
"tableRelation"); //$NON-NLS-1$ //$NON-NLS-2$
        ConnectorObject firstConnector = new
ConnectorObject(Cardinality.getValue("(0,n)")); //$NON-NLS-1$
        objects[1] = graph.insertEdge(graph.getDefaultParent(), null, firstConnector,
source, objects[2], "entityRelationConnector"); //$NON-NLS-1$
        ConnectorObject secondConnector = new
ConnectorObject(Cardinality.getValue("(0,n)")); //$NON-NLS-1$
        objects[3] = graph.insertEdge(graph.getDefaultParent(), null, secondConnector,
objects[2], target, "entityRelationConnector"); //$NON-NLS-1$
        break;

    case ERROR:
        JOptionPane.showMessageDialog((JFrame)
SwingUtilities.getWindowForComponent(mainWindow),
        "A conexão entre estes objetos não pode existir", "Aviso", 0);
        //$NON-NLS-1$
    }

    if (!(connection == Connection.NONE || connection ==
Connection.ENTITY_TO_INHERITANCE ||
        connection == Connection.ERROR))
        AdjustEdgeLabelPosition(objects);
}

public void AdjustEdgeLabelPosition(Object[] objects) {
    mxGraph graph = modelingComponent.getGraph();
    mxIGraphModel model = graph.getModel();
    int numConectors = objects.length > 3 ? 2 : 1;

```

```

int indexConector = 1;
int indexSource = 0;
while (numConectors > 0) {
    mxCellState state = graph.getView()
        .getState(objects[indexConector]);
    List<mxPoint> points = NearestPointsFromSource(
        objects[indexSource], state);
    mxPoint firstPoint = points.get(0);
    mxPoint secondPoint = points.get(1);
    double newX = firstPoint.getX()
        + ((secondPoint.getX() - firstPoint.getX()) / 2);
    double newY = firstPoint.getY()
        + ((secondPoint.getY() - firstPoint.getY()) / 2);
    mxGeometry geometry = model.getGeometry(state.getCell());

    if (geometry != null) {
        geometry = (mxGeometry) geometry.clone();

        // Resets the relative location stored inside the geometry
        mxPoint pt = graph.getView()
            .getRelativePoint(state, newX, newY);
        geometry.setX(pt.getX());
        geometry.setY(pt.getY());

        // Resets the offset inside the geometry to find the offset
        // from the resulting point
        double scale = graph.getView().getScale();
        geometry.setOffset(new mxPoint(0, 0));
        pt = graph.getView().getPoint(state, geometry);
        geometry.setOffset(new mxPoint(Math.round((newX - pt.getX())
            / scale), Math.round((newY - pt.getY()) / scale)));

        model.setGeometry(state.getCell(), geometry);
    }
    numConectors--;
    indexConector += 2;
    indexSource += 4;
}
}

public List<mxPoint> NearestPointsFromSource(Object object,
    mxCellState state) {
    List<mxPoint> points = new ArrayList<mxPoint>();
    mxICell sourceCell = (mxICell) object;
    double srcX = sourceCell.getGeometry().getX();
    double firstPoint = state.getAbsolutePoints().get(0).getX();
    double lastPoint = state.getAbsolutePoints().get(3).getX();
    if (Math.abs(firstPoint - srcX) < Math.abs(lastPoint - srcX)) {
        points.add(state.getAbsolutePoints().get(0));
        points.add(state.getAbsolutePoints().get(1));
    } else {
        points.add(state.getAbsolutePoints().get(2));
        points.add(state.getAbsolutePoints().get(3));
    }

    return points;
}
}
}

package ufsc.sisinf.br.modelo2all.control;

import java.awt.BorderLayout;
import java.awt.Frame;

```



```

import javax.swing.JDialog;
import javax.swing.JTextArea;

import com.mxgraph.util.mxResources;

public class SqlEditor extends JDialog {

    /**
     *
     */
    private static final long serialVersionUID = -5951065346955910406L;

    private JTextArea textArea = new JTextArea();

    public SqlEditor(Frame owner) {
        super(owner);

        setTitle(mxResources.get("physicalModeling"));
        setLayout(new BorderLayout());
        setSize(800, 600);

        getContentPane().add(textArea);
    }

    public void insertSqlInstruction(String instruction) {
        String newText = textArea.getText() + "\n" + instruction;
        textArea.setText(newText);
    }
}

package ufsc.sisinf.brmodelo2all.control;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.SwingUtilities;

import com.mxgraph.model.mxCell;
import com.mxgraph.model.mxGeometry;
import com.mxgraph.util.mxResources;

import ufsc.sisinf.brmodelo2all.model.Cardinality;
import ufsc.sisinf.brmodelo2all.model.Modeling;
import ufsc.sisinf.brmodelo2all.model.ModelingComponent;
import ufsc.sisinf.brmodelo2all.model.objects.AttributeObject;
import ufsc.sisinf.brmodelo2all.model.objects.ColumnObject;
import ufsc.sisinf.brmodelo2all.model.objects.ConnectorObject;
import ufsc.sisinf.brmodelo2all.model.objects.EntityObject;
import ufsc.sisinf.brmodelo2all.model.objects.InheritanceObject;
import ufsc.sisinf.brmodelo2all.model.objects.ModelingObject;
import ufsc.sisinf.brmodelo2all.model.objects.RelationObject;
import ufsc.sisinf.brmodelo2all.model.objects.TableObject;
import ufsc.sisinf.brmodelo2all.ui.AppMainWindow;
import ufsc.sisinf.brmodelo2all.ui.ConversionIteratorWindow;

public class ConceptualConversor {

    private final ModelingComponent conceptualModelingComponent;

```

```

private final ModelingEditor logicalModelingEditor;

private final AppMainWindow mainWindow;

private List<TableObject> tablesCreated = new ArrayList<TableObject>();

@SuppressWarnings("unused")
private boolean adoptSugestions = false;

private boolean canceledByUser = false;

private boolean tablesForAllEntities = false;

private boolean noTablesForAllEntities = false;

public ConceptualConversor(ModelingComponent conceptualModelingComponent,
ModelingEditor logicalModelingEditor, final AppMainWindow mainWindow) {
    this.conceptualModelingComponent = conceptualModelingComponent;
    this.logicalModelingEditor = logicalModelingEditor;
    this.mainWindow = mainWindow;
}

public void convertModeling() {
    // if (not ExportarEntidades(oModelo)) OK
    // or (not ExportarAtributos(oModelo)) OK
    // or (not ExportarAutoRelacionamento(oModelo)) OK
    // or (not ExportarEspecializacoes(oModelo)) OK_PARCIAL
    // or (not ExportarEntidadesHerancaMultipla(oModelo))
    // or (not ExportarRelacoes(oModelo))
    // or (not ExportarEntAss(oModelo))
    // or (not ExportarTextos(oModelo))
    // or (not ExportarLimparLixo(oModelo))
    List<mxCell> inheritanceObjects = new ArrayList<mxCell>();
    List<mxCell> relationObjects = new ArrayList<mxCell>();
    // run through all entities at the modeling and transform each one in a table for the logical
modeling
    // check if the entity is weak to deal with its conversion the way it is supposed to be dealed
Object[] cells =
conceptualModelingComponent.getCells(conceptualModelingComponent.getBounds());
    for (int i = 0; i < cells.length && !canceledByUser; i++) {
        Object cell = cells[i];
        if (cell instanceof mxCell) {
            mxCell objectCell = (mxCell)cell;
            mxGeometry geometry = objectCell.getGeometry();
            if (objectCell.getValue() instanceof EntityObject) {
                EntityObject entity = (EntityObject) objectCell.getValue();
                convertEntity(entity, geometry);
                weakEntityConversion(objectCell, geometry);
                if (entity.isSelfRelated())
                    convertSelfRelatedEntity(objectCell, geometry);

            } else if (objectCell.getValue() instanceof InheritanceObject)
                inheritanceObjects.add(objectCell);
            else if (objectCell.getValue() instanceof RelationObject)
                relationObjects.add(objectCell);
        }
    }

    if (canceledByUser){
        JOptionPane.showMessageDialog((JFrame)
SwingUtilities.getWindowForComponent(mainWindow),
"Conversão cancelada pelo usuário", "Aviso", 0); //$NON-NLS-
1$

return;

```

```

    }

    // after converting all entities, deal with inheritances and relations conversion
    List<mxCell> orderedInheritanceObjects =
orderParentInheritanceList(inheritanceObjects);
    if (inheritanceObjects.size() > 0 && orderedInheritanceObjects.size() == 0){
        JOptionPane.showMessageDialog((JFrame)
SwingUtilities.getWindowForComponent(mainWindow),
        "Referência circular na especialização/generalização de
entidades! Clique em OK para finalizar a conversão.", "Aviso", 0); //$NON-NLS-1$
        return;
    }
    orderChildrenInheritanceList(inheritanceObjects, orderedInheritanceObjects);

    Iterator<mxCell> iterator = orderedInheritanceObjects.iterator();
    while (iterator.hasNext() && !canceledByUser) {
        convertInheritances(iterator.next());
    }

    if (canceledByUser){
        JOptionPane.showMessageDialog((JFrame)
SwingUtilities.getWindowForComponent(mainWindow),
        "Conversão cancelada pelo usuário", "Aviso", 0); //$NON-NLS-
1$
        return;
    }

    iterator = relationObjects.iterator();
    while (iterator.hasNext() && !canceledByUser) {
        convertRelations(iterator.next());
    }

    if (canceledByUser)
        JOptionPane.showMessageDialog((JFrame)
SwingUtilities.getWindowForComponent(mainWindow),
        "Conversão cancelada pelo usuário", "Aviso", 0); //$NON-NLS-
1$
    }

    public void convertEntity(EntityObject entity, mxGeometry geometry){
        String tableName = entity.getName();
        double x = geometry.getX();
        double y = geometry.getY();

        Modeling logicalModeling =
(Modeling)logicalModelingEditor.modelingComponent.getGraph();

        Object table = logicalModeling.insertVertex(logicalModeling.getDefaultParent(), null,
        new TableObject(tableName), x, y, 150, 30, //$NON-NLS-1$
        "table"); //$NON-NLS-1$

        tablesCreated.add((TableObject)((mxCell)table).getValue());

        // check for attributes for this entity and convert them as fields for the created table
        int count = entity.getChildObjects().size();
        for (int i = 0; i < count && !canceledByUser; i++) {
            mxCell cell = (mxCell)entity.getChildObjects().get(i);
            if (cell.getValue() instanceof AttributeObject)
                convertAttribute((AttributeObject)cell.getValue(),
((mxCell)table).getGeometry());
        }
    }

    public void convertAttribute(AttributeObject attribute, mxGeometry tableGeometry) {

```

```

int selectedAlternative = 1;
List<AttributeObject> attributes = new ArrayList<AttributeObject>();
double x = tableGeometry.getCenterX();
double y = tableGeometry.getCenterY();

    if ((attribute.isMultiValued() || attribute.isComposed()) &&
        !(attribute.isMultiValued() && attribute.getMaximumCardinality() ==
'1' &&
            !attribute.isComposed())) {
        String[] messages = new String[2];
        String[] alternatives = new String[3];
        fillAttributeInteraction(attribute, messages, alternatives, 1/*situation*/);
        int suggestionIndex = Integer.parseInt(new
Character(attribute.getMaximumCardinality().toString()) > 1 ?
                                0 : 1;
        selectedAlternative = askToUser(messages, alternatives, suggestionIndex);

        getAttributeChilds(attribute, attributes);

    } else {
        attributes.add(attribute);
        if (attribute.isOptional()) {
            String[] messages = new String[2];
            String[] alternatives = new String[3];
            fillAttributeInteraction(attribute, messages, alternatives, 2/*situation*/);

            selectedAlternative = askToUser(messages, alternatives,
selectedAlternative);
        }
    }

    if (selectedAlternative == -1) {
        canceledByUser = true;
        return;
    }

    if (selectedAlternative == 0) {
        // criar uma tabela nova com o nome do atributo
        Modeling logicalModeling =
(Modeling)logicalModelingEditor.modelingComponent.getGraph();

        mxCell tableCell =
(mxCell)logicalModeling.insertVertex(logicalModeling.getDefaultParent(), null,
        new TableObject(attribute.getName()),
tableGeometry.getCenterX() + (tableGeometry.getWidth() * 2),
        tableGeometry.getCenterY(), 150, 30, "table"); //$NON-NLS-1$

        tablesCreated.add((TableObject)((mxCell)tableCell).getValue());

        mxGeometry newTableGeometry = tableCell.getGeometry();
        x = newTableGeometry.getCenterX();
        y = newTableGeometry.getCenterY();
    }

    Iterator<AttributeObject> iterator = attributes.iterator();
    while (iterator.hasNext()) {
        AttributeObject attributeObject = iterator.next();
        String style = attributeObject.isIdentifier() ? mxResources.get("primaryKey") :
"column";

        Object column =
logicalModelingEditor.modelingManager.insertColumnObject(x, y, style);
        ColumnObject columnObject = (ColumnObject)((mxCell)column).getValue();
        columnObject.setName(attributeObject.getName());
        columnObject.setType(attributeObject.getType());
    }

```



```

relationObject =
(RelationObject)((mxCell)edge.getTarget()).getValue();
else
relationObject =
(RelationObject)((mxCell)edge.getSource()).getValue();

if (relationObject.getRelatedObjects().size() == 2) {
ModelingObject otherEntity =
relationObject.getRelatedObjects().get(0) == (ModelingObject)entityCell.getValue() ?

relationObject.getRelatedObjects().get(1) :

relationObject.getRelatedObjects().get(0);

Iterator<Object> iterator =
otherEntity.getChildObjects().iterator();

while (iterator.hasNext()) {
Object object = iterator.next();
if (((mxCell)object).getValue() instanceof
AttributeObject) {
AttributeObject attributeObject =
(AttributeObject)((mxCell)object).getValue();

if (attributeObject.isIdentifier()) {
Object column =
logicalModelingEditor.modelingManager.insertColumnObject(x + 75, y + 15, "primaryKey");
ColumnObject
columnObject = (ColumnObject)((mxCell)column).getValue();

columnObject.setName(attributeObject.getName());

columnObject.setType(attributeObject.getType());

columnObject.setForeignKey(true);
}
}
}
}
}
}
}

public void convertSelfRelatedEntity(mxCell entityObject, mxGeometry geometry) {
EntityObject entity = (EntityObject)entityObject.getValue();
Iterator<RelationObject> iterator = entity.getRelationsIterator();
while (iterator.hasNext()) {
RelationObject relation = iterator.next();
Iterator<ModelingObject> iterator2 = relation.getRelatedObjectsIterator();
int count = 0;
while (iterator2.hasNext() && count < 2) {
if (iterator2.next() == entity)
count++;
}

if (count == 2) {
int edgeCount = entityObject.getEdgeCount();
Cardinality minorCardinality = null;
Cardinality majorCardinality = null;
for (int i = 0; i < edgeCount; i++) {
mxCell edge = (mxCell) entityObject.getEdgeAt(i);

if (edge.getValue() instanceof ConnectorObject &&
(edge.getSource().getValue() == entity ||
edge.getSource().getValue() == relation) &&

```

```

edge.getTarget().getValue() == relation)) {
    (edge.getTarget().getValue() == entity ||
ConnectorObject connector =
    (ConnectorObject)edge.getValue();
    if (minorCardinality == null)
        minorCardinality =
connector.getCardinality();
    else {
        if
(Cardinality.minor(connector.getCardinality(), minorCardinality)) {
            majorCardinality = minorCardinality;
            minorCardinality =
connector.getCardinality();
        } else
            majorCardinality =
connector.getCardinality();
    }
}
}

boolean addNewColumn;
if (majorCardinality == Cardinality.ZERO_ONE){
    addNewColumn = true;
} else if (minorCardinality == Cardinality.ONE_N || minorCardinality ==
Cardinality.ZERO_N){
    addNewColumn = false;
} else {
    String[] messages = new String[2];
    String[] alternatives = new String[3];
    fillSelfRelatedInteraction(entity, relation, minorCardinality,
majorCardinality, messages, alternatives);
    int suggestionIndex = majorCardinality ==
Cardinality.ZERO_ONE ? 0 : 1;
    int selectedAlternative = askToUser(messages, alternatives,
suggestionIndex);
    addNewColumn = selectedAlternative == 0;
}

double x = geometry.getCenterX();
double y = geometry.getCenterY();
mxGeometry tableGeometry = geometry;
if (addNewColumn) {
    int num = entity.getChildObjects().size();
    for (int i = 0; i < num; i++) {
        mxCell cell = (mxCell)entity.getChildObjects().get(i);
        if (cell.getValue() instanceof AttributeObject &&
((AttributeObject)cell.getValue()).isIdentifier()) {
            AttributeObject attributeObject =
(AttributeObject)cell.getValue();
            Object column =
logicalModelingEditor.modelingManager.insertColumnObject(x, y, "column");
            ColumnObject columnObject =
(ColumnObject)((mxCell)column).getValue();
            columnObject.setName("possui" +
attributeObject.getName());
            columnObject.setType(attributeObject.getType());
        }
    }
} else {
    // create tables and generate fields
    String tableName = relation.getName();

```

```

x *= 1.5;

Modeling logicalModeling =
(Modeling)logicalModelingEditor.modelingComponent.getGraph();

Object table =
logicalModeling.insertVertex(logicalModeling.getDefaultParent(), null,
new TableObject(tableName), x, y, 150, 30,
//$NON-NLS-1$
"table"); //$NON-NLS-1$
tablesCreated.add((TableObject)((mxCell)table).getValue());
tableGeometry = ((mxCell)table).getGeometry();
x = tableGeometry.getCenterX();
y = tableGeometry.getCenterY();

int num = entity.getChildObjects().size();
for (int i = 0; i < num; i++) {
    mxCell cell = (mxCell)entity.getChildObjects().get(i);
    if (cell.getValue() instanceof AttributeObject &&
((AttributeObject)cell.getValue()).isIdentifier()) {
        AttributeObject attributeObject =
(AttributeObject)cell.getValue();
        Object column =
logicalModelingEditor.modelingManager.insertColumnObject(x, y, "column");
        ColumnObject columnObject =
(ColumnObject)((mxCell)column).getValue();
        columnObject.setName("possui" +
attributeObject.getName());
        columnObject.setType(attributeObject.getType());
    }
}

// self related attributes
int num = relation.getChildObjects().size();
for (int i = 0; i < num; i++) {
    mxCell cell = (mxCell)relation.getChildObjects().get(i);
    if (cell.getValue() instanceof AttributeObject)
        convertAttribute((AttributeObject)cell.getValue(),
tableGeometry);
}
}
}

public void fillSelfRelatedInteraction(EntityObject entity, RelationObject relation, Cardinality
minorCardinality, Cardinality majorCardinality, String[] messages, String[] alternatives) {
    messages[0] = "O que fazer a respeito do auto-relacionamento com cardinalidade ";
    messages[0] += Cardinality.getText(minorCardinality) + " e ";
    messages[0] += Cardinality.getText(majorCardinality);
    messages[1] = "encontrado entre a entidade \"" + entity.getName() + "\" e o
relacionamento \"";
    messages[1] += relation.getName() + "\"?";

    alternatives[0] = "1) Criar relacionamento recursivo em \"" + entity.getName() + "\"";
    alternatives[1] = "2) Criar uma tabela para o auto-relacionamento";
    alternatives[2] = "3) Deste ponto em diante aceitar todas as sugestões";
}

public List<mxCell> orderParentInheritanceList(List<mxCell> inheritanceObjects) {
    List<mxCell> orderedInheritanceObjects = new ArrayList<mxCell>();

```



```

        Iterator<mxCell> iterator = inheritanceObjects.iterator();
        while (iterator.hasNext()) {
            mxCell inheritanceCell = iterator.next();
            InheritanceObject inheritanceObject =
(InheritanceObject)inheritanceCell.getValue();
            EntityObject entityObject =
(EntityObject)((mxCell)inheritanceObject.getParentObject()).getValue();
            mxCell cell = (mxCell)entityObject.getParentObject();
            if (cell == null || !(cell.getValue() instanceof InheritanceObject))
                orderedInheritanceObjects.add(inheritanceCell);
        }

        return orderedInheritanceObjects;
    }

    public void orderChildrenInheritanceList(List<mxCell> inheritanceObjects, List<mxCell>
orderedInheritanceObjects){
        Iterator<mxCell> iterator = orderedInheritanceObjects.iterator();
        while (iterator.hasNext()) {
            mxCell inheritanceCell = iterator.next();
            getInheritanceChildren(inheritanceCell, orderedInheritanceObjects);
        }
    }

    public void getInheritanceChildren(mxCell inheritanceCell, List<mxCell>
orderedInheritanceObjects) {
        InheritanceObject inheritanceObject = (InheritanceObject)inheritanceCell.getValue();
        Iterator<Object> iterator = inheritanceObject.getChildObjects().iterator();
        while (iterator.hasNext()) {
            mxCell entityCell = (mxCell)iterator.next();
            EntityObject entityObject = (EntityObject)entityCell.getValue();
            Iterator<Object> iterator2 = entityObject.getChildObjects().iterator();
            while (iterator2.hasNext()) {
                mxCell cell = (mxCell)iterator2.next();
                if (cell.getValue() instanceof InheritanceObject) {
                    if (orderedInheritanceObjects.indexOf(cell) > -1)
                        orderedInheritanceObjects.remove(cell);

                    orderedInheritanceObjects.add(cell);

                    getInheritanceChildren(cell, orderedInheritanceObjects);
                }
            }
        }
    }

    public void convertInheritances(mxCell inheritanceCell) {
        InheritanceObject inheritanceObject = (InheritanceObject)inheritanceCell.getValue();
        if (inheritanceObject.getChildObjects().size() < 1)
            return;
        String[] messages = new String[2];
        String[] alternatives = new String[!inheritanceObject.ispartial() ? 4 : 3];

        fillInheritanceInteraction(inheritanceObject, messages, alternatives);

        int answer = askToUser(messages, alternatives, inheritanceObject.ispartial() ? 0 :
1/*suggestionIndex*/);
        if (answer == 2) {
            if (inheritanceObject.getChildObjects().size() > 1) {
                EntityObject entityObject =
(EntityObject)((mxCell)inheritanceObject.getParentObject()).getValue();

```

```

        JOptionPane.showMessageDialog((JFrame)
SwingUtilities.getWindowForComponent(mainWindow),
        "A tabela \" + entityObject.getName() + "\" não será
excluída pois possui herança múltipla", "Aviso", 0); //$NON-NLS-1$
        answer = -1;
    } else {
        EntityObject entityObject =
(EntityObject)((mxCell)inheritanceObject.getParentObject()).getValue();
        Iterator<RelationObject> iterator = entityObject.getRelationsIterator();
        if (iterator.hasNext()) {
            JOptionPane.showMessageDialog((JFrame)
SwingUtilities.getWindowForComponent(mainWindow),
        "A tabela \" + entityObject.getName() + "\"
não será excluída pois possui pelo menos um relacionamento", "Aviso", 0); //$NON-NLS-1$
            answer = -1;
        }
    }
    if (answer == -1) {
        messages[0] = "[REFAZER] " + messages[0];

        alternatives = new String[2];
        alternatives[0] = "1) Criar uma tabela para cada entidade";
        alternatives[1] = "2) Criar uma única tabela para toda a hierarquia";

        answer = askToUser(messages, alternatives, inheritanceObject.ispartial()
? 0 : 1/*suggestionIndex*/);
    }
}

if (answer == -1) {
    canceledByUser = true;
    return;
}

if (answer == 0) { // one table for each entity
    int count = inheritanceObject.getChildObjects().size();
    for (int i = 0; i < count && !canceledByUser; i++) {
        mxCell cell = (mxCell)inheritanceObject.getChildObjects().get(i);
        if (cell.getValue() instanceof EntityObject) {
            EntityObject entityObject = (EntityObject)cell.getValue();
//           //se tem múltiplas origens então já foi processado com uma tabela única
//           if Ent.ehHerancaMultipla then Continue;
            TableObject tableObject =

tableByName(entityObject.getName());
            if (tableObject != null) {
                if (tableObject.getChildObjects().size() == 0) {
                    int suggestionIndex = 0;
                    int otherAnswer = 0;
                    if (tablesForAllEntities)
                        suggestionIndex = 3;
                    else if (noTablesForAllEntities)
                        suggestionIndex = 2;
                    else {
                        messages = new String[3];
                        messages[0] = "O que fazer a respeito
da entidade ";
                        entityObject.getName() + "\" ";
                        + inheritanceObject.getName() + "\" do tipo ";
                        inheritanceObject.ispartial() ? "parcial e " : "total e ";
                        inheritanceObject.isExclusive() ? "exclusiva" : "opcional";
                        messages[0] += "\" +
                        messages[1] += "especializada de \""
                        messages[1] +=
                        messages[1] +=
                    }
                }
            }
        }
    }
}

```

```

atributos?";

tabela para a entidade (Heuser)";
a entidade";
entidade nestas condições (Heuser)";
nestas condições";

alternatives, suggestionIndex);

messages[2] += "que não possui

alternatives = new String[4];
alternatives[0] = "1) NÃO gerar

alternatives[1] = "2) Gerar tabela para

alternatives[2] = "3) NÃO gerar tabela para nenhuma

alternatives[3] = "4) Gerar tabela para todas as entidade

otherAnswer = askToUser(messages,

if (otherAnswer == -1) {
    canceledByUser = true;
    return;
}
tablesForAllEntities = otherAnswer == 3;
noTablesForAllEntities = otherAnswer == 2;

}

if (otherAnswer == 0 || otherAnswer == 2){
    pegar todas as ligações de tableObject
    - remover a ligação corrente de
    - adicionar a ligação na tabela pai

continue;

}

}

//
// iterativamente
//
// tableObject
//
// (entidade pai da especialização)

//
// especialização)
//
// campos que são chaves na tabela pai

//
// caso a tabela pai seja mantida, ligar as tabelas filho nela
// sempre a cardinalidade em relação a TabelaP é (0,n)

}
}

} else if (answer == 1) { // one table for the whole hierarchy
    int count = inheritanceObject.getChildObjects().size();
    for (int i = 0; i < count && !canceledByUser; i++) {
        mxCell cell = (mxCell)inheritanceObject.getChildObjects().get(i);
        if (cell.getValue() instanceof EntityObject) {
            EntityObject entityObject = (EntityObject)cell.getValue();

            //
            // if Ent.ehHerancaMultipla then
            //
            // JOptionPane.showMessageDialog((JFrame)
            SwingUtilities.getWindowForComponent(mainWindow),
            //
            // "Herança múltipla encontrada! Não será
            possível realizar a mesclagem em uma única tabela.", "Aviso", 0); //$NON-NLS-1$
            //
            // return;
            TableObject tableObject =
tableByName(entityObject.getName());

```

```

        if (tableObject != null) {
//          pegar todas as ligações de tableObject iterativamente
//          - remover a ligação corrente de tableObject
//          - adicionar a ligação na tabela pai (entidade pai da
especialização)

//          pegar todos os campos de tableobject iterativamente
//          e adicionar na tabela pai (entidade pai da
especialização)
//          modificar o nome da tabela pai para "tabela
pai+tableObject"
        }
    }
}

} else if (answer == 2) { // table only for the specialized entity
// //se for parcial não excluir.
// if E.Parcial then
//     TabelaP.Interferencia := TabelaP.Interferencia + 1;
//     int count = inheritanceObject.getChildObjects().size();
//     for (int i = 0; i < count && !canceledByUser; i++) {
//         mxCell cell = (mxCell)inheritanceObject.getChildObjects().get(i);
//         if (cell.getValue() instanceof EntityObject) {
//             EntityObject entityObject = (EntityObject)cell.getValue();
//             @SuppressWarnings("unused")
//             TableObject tableObject =
tableByName(entityObject.getName());

//             iterar pelos campos da tabela pai para copiar para a tableObject

//             excluir tabela pai
        }
    }
}

}

}

public void fillInheritanceInteraction(InheritanceObject inheritanceObject, String[] messages,
String[] alternatives){
    messages[0] = "O que fazer a respeito da especialização ";
    messages[0] += inheritanceObject.ispartial() ? "parcial e " : "total e ";
    messages[0] += inheritanceObject.isExclusive() ? "exclusiva " : "opcional ";
    messages[0] += "\"" + inheritanceObject.getName() + "\"";
    messages[1] = "encontrada partindo da entidade \"";
    messages[1] +=
((ModelingObject)((mxCell)(inheritanceObject.getParentObject()).getValue())).getName() + "\"?";

    alternatives[0] = "1) Criar uma tabela para cada entidade";
    alternatives[1] = "2) Criar uma única tabela para toda a hierarquia";
    if (!inheritanceObject.ispartial()) {
        alternatives[2] = "3) Criar tabela(s) apenas para a(s) entidade(s) especializada(s)";
        alternatives[3] = "4) Deste ponto em diante aceitar todas as sugestões";
    } else
        alternatives[2] = "3) Deste ponto em diante aceitar todas as sugestões";
}

}

public void convertRelations(mxCell relationCell) {
    RelationObject relationObject = (RelationObject)relationCell.getValue();
    if (relationObject.getRelatedObjects().size() < 2) {
        JOptionPane.showMessageDialog((JFrame)
SwingUtilities.getWindowForComponent(mainWindow),

```

"O relacionamento \" + relationObject.getName() + "\" não foi bem formado e será desconsiderado nesta conversão!", "Aviso", 0); //\$NON-NLS-1\$

```
        return;
    }

    List<EntityObject> entitiesList = new ArrayList<EntityObject>();
    boolean noAdd = false;
    Cardinality maximumCardinality = Cardinality.ONE_ONE;
    int relatedObjectsCount = relationObject.getRelatedObjects().size();
    for (int i = 0; i < relatedObjectsCount; i++) {
        ModelingObject object = relationObject.getRelatedObjects().get(i);
        if (object instanceof EntityObject) {
            EntityObject entityObject = (EntityObject)object;
            ConnectorObject conectorObject =
relationObject.getConnectorObject(relationCell, entityObject);
            Cardinality insideMaximumCardinality =
conectorObject.getCardinality();
            noAdd = false;
            for (int j = i + 1; j < relatedObjectsCount; j++) {
relationObject.getRelatedObjects().get(j);
                if (object2 instanceof EntityObject) {
                    EntityObject entityObject2 = (EntityObject)object2;
                    ConnectorObject conectorObject2 =
relationObject.getConnectorObject(relationCell, entityObject2);
                    if (!Cardinality.major(insideMaximumCardinality,
conectorObject2.getCardinality()))
                        insideMaximumCardinality =
conectorObject2.getCardinality();
                    noAdd = true;
                    break;
                }
            }
            if (!Cardinality.major(maximumCardinality,
insideMaximumCardinality))
                maximumCardinality = insideMaximumCardinality;
            if (!noAdd && entitiesList.indexOf(entityObject) == -1)
                entitiesList.add(entityObject);
        }
    }

    if (entitiesList.size() < 2)
        return;

    if (entitiesList.size() > 2) {
        boolean minorCardinality = true;
        relatedObjectsCount = relationObject.getRelatedObjects().size();
        for (int i = 0; i < relatedObjectsCount && minorCardinality; i++) {
            ModelingObject object = relationObject.getRelatedObjects().get(i);
            if (object instanceof EntityObject) {
                EntityObject entityObject = (EntityObject)object;
                ConnectorObject conectorObject =
relationObject.getConnectorObject(relationCell, entityObject);
                minorCardinality = conectorObject.getCardinality() ==
Cardinality.ONE_ONE;
            }
        }
        if (minorCardinality) {
    }
```

```

        } else {

        }

    }

    public int askToUser(String[] messages, String[] alternatives, int suggestionIndex) {
        JFrame frame = (JFrame) SwingUtilities.getWindowForComponent(mainWindow);

        ConversionIteratorWindow conversionIterator = new ConversionIteratorWindow(frame,
messages, alternatives, suggestionIndex);
        conversionIterator.setModal(true);

        // Centers inside the application frame
        int x = frame.getX() + (frame.getWidth() - conversionIterator.getWidth()) / 2;
        int y = frame.getY() + (frame.getHeight() - conversionIterator.getHeight()) / 2;
        conversionIterator.setLocation(x, y);

        // Shows the modal dialog and waits
        conversionIterator.setVisible(true);

        return conversionIterator.getResult();
    }

    private TableObject tableByName(String tableName) {
        TableObject table = null;
        Iterator<TableObject> iterator = tablesCreated.iterator();
        while (iterator.hasNext() && table == null) {
            TableObject tableObject = iterator.next();
            if (tableObject.getName() == tableName)
                table = tableObject;
        }
        return table;
    }
}
package ufsc.sisinf.br.modelo2all.ui;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.Frame;
import java.awt.GradientPaint;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;

import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JComponent;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRootPane;
import javax.swing.KeyStroke;

import com.mxgraph.util.mxResources;
import com.mxgraph.view.mxGraph;

public class AboutWindow extends JDialog {

    /**
     *

```

```

*/
private static final long serialVersionUID = -3378029138434324390L;

/**
 *
 */
public AboutWindow(Frame owner) {
    super(owner);
    setTitle(mxResources.get("aboutGraphEditor"));
    setLayout(new BorderLayout());

    // Creates the gradient panel
    JPanel panel = new JPanel(new BorderLayout()) {

        /**
         *
         */
        private static final long serialVersionUID = -5062895855016210947L;

        /**
         *
         */
        public void paintComponent(Graphics g) {
            super.paintComponent(g);

            // Paint gradient background
            Graphics2D g2d = (Graphics2D) g;
            g2d.setPaint(new GradientPaint(0, 0, Color.WHITE, getWidth(), 0,
getBackground());
            g2d.fillRect(0, 0, getWidth(), getHeight());
        }
    };

    panel.setBorder(BorderFactory.createCompoundBorder(BorderFactory
        .createMatteBorder(0, 0, 1, 0, Color.GRAY), BorderFactory
        .createEmptyBorder(8, 8, 12, 8)));

    // Adds title
    JLabel titleLabel = new JLabel(mxResources.get("aboutGraphEditor"));
    titleLabel.setFont(titleLabel.getFont().deriveFont(Font.BOLD));
    titleLabel.setBorder(BorderFactory.createEmptyBorder(4, 0, 0, 0));
    titleLabel.setOpaque(false);
    panel.add(titleLabel, BorderLayout.NORTH);

    // Adds optional subtitle
    JLabel subtitleLabel = new JLabel(
        "Para maiores informações visite https://code.google.com/p/brmodelonext/");
    subtitleLabel.setBorder(BorderFactory.createEmptyBorder(4, 18, 0, 0));
    subtitleLabel.setOpaque(false);
    panel.add(subtitleLabel, BorderLayout.CENTER);

    getContentPane().add(panel, BorderLayout.NORTH);

    JPanel content = new JPanel();
    content.setLayout(new BoxLayout(content, BoxLayout.Y_AXIS));
    content.setBorder(BorderFactory.createEmptyBorder(12, 12, 12, 12));

    content.add(new JLabel("brModelo2All - Modelagem Lógica/Conceitual descomplicada
para todos"));
    content.add(new JLabel(""));

    content.add(new JLabel("Versão brModeloNext " + mxGraph.VERSION));
    content.add(new JLabel(""));
    content.add(new JLabel(""));
    content.add(new JLabel(""));
}

```

```

try
{
    content.add(new JLabel("Sistema Operacional: "
        + System.getProperty("os.name")));
    content.add(new JLabel("Versão do Sistema Operacional: "
        + System.getProperty("os.version")));
    content.add(new JLabel(""));

    content.add(new JLabel("Java Vendor: "
        + System.getProperty("java.vendor", "undefined")));
    content.add(new JLabel("Java Version: "
        + System.getProperty("java.version", "undefined")));
    content.add(new JLabel(""));

    content.add(new JLabel("Memória Total: "
        + Runtime.getRuntime().totalMemory()));
    content.add(new JLabel("Memória Livre: "
        + Runtime.getRuntime().freeMemory()));
}
catch (Exception e)
{
    // ignore
}

getContentPane().add(content, BorderLayout.CENTER);

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));
buttonPanel.setBorder(BorderFactory.createCompoundBorder(BorderFactory
    .createMatteBorder(1, 0, 0, 0, Color.GRAY), BorderFactory
    .createEmptyBorder(16, 8, 8, 8)));
getContentPane().add(buttonPanel, BorderLayout.SOUTH);

// Adds OK button to close window
JButton closeButton = new JButton("Fechar");
closeButton.addActionListener(new ActionListener()
{
    public void actionPerformed(ActionEvent e)
    {
        setVisible(false);
    }
});

buttonPanel.add(closeButton);

// Sets default button for enter key
getRootPane().setDefaultButton(closeButton);

setResizable(false);
setSize(400, 400);
}

/**
 * Overrides {@link JDialog#createRootPane()} to return a root pane that
 * hides the window when the user presses the ESCAPE key.O
 */
protected JRootPane createRootPane() {
    KeyStroke stroke = KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0);
    JRootPane rootPane = new JRootPane();
    rootPane.registerKeyboardAction(new ActionListener()
    {
        public void actionPerformed(ActionEvent actionEvent)
        {
            setVisible(false);
        }
    });
}

```



```

        }, stroke, JComponent.WHEN_IN_FOCUSED_WINDOW);
        return rootPane;
    }
}

```

```
package ufsc.sisinf.brmodelo2all.util;
```

```
public class AppConstants {

    public static final int TEXT_FIELD = 0;

    public static final int CHECK_BOX = 1;

    public static final int COMBO_BOX = 2;

}

```

```
package ufsc.sisinf.brmodelo2all.util;
```

```
import java.io.File;

import javax.imageio.ImageIO;
import javax.swing.JFileChooser;
import javax.swing.filechooser.FileFilter;

/**
 * Filter for use in a { @link JFileChooser}.
 */
public class AppDefaultFileFilter extends FileFilter {

```

```

    /**
     * Extension of accepted files.
     */
    protected String ext;

    /**
     * Description of accepted files.
     */
    protected String desc;

    /**
     * Constructs a new filter for the specified extension and description.
     *
     * @param extension
     *     The extension to accept files with.
     * @param description
     *     The description of the file format.
     */
    public AppDefaultFileFilter(String extension, String description)
    {
        ext = extension.toLowerCase();
        desc = description;
    }

```

```

    /**
     * Returns true if <code>file</code> is a directory or ends with
     * { @link #ext}.
     *
     * @param file
     *     The file to be checked.
     * @return Returns true if the file is accepted.
     */
    public boolean accept(File file)

```

```

{
    return file.isDirectory() || file.getName().toLowerCase().endsWith(ext);
}

/**
 * Returns the description for accepted files.
 *
 * @return Returns the description.
 */
public String getDescription()
{
    return desc;
}

/**
 * Returns the extension for accepted files.
 *
 * @return Returns the extension.
 */
public String getExtension()
{
    return ext;
}

/**
 * Sets the extension for accepted files.
 *
 * @param extension
 *     The extension to set.
 */
public void setExtension(String extension)
{
    this.ext = extension;
}

/**
 * Utility file filter to accept all image formats supported by image io.
 *
 * @see ImageIO#getReaderFormatNames()
 */
public static class ImageFileFilter extends FileFilter
{

    /**
     * Holds the accepted file format extensions for images.
     */
    protected static String[] imageFormats = ImageIO.getReaderFormatNames();

    /**
     * Description of the filter.
     */
    protected String desc;

    /**
     * Constructs a new file filter for all supported image formats using
     * the specified description.
     *
     * @param description
     *     The description to use for the file filter.
     */
    public ImageFileFilter(String description)
    {
        desc = description;
    }
}

```

```

/**
 * Returns true if the file is a directory or ends with a known image
 * extension.
 *
 * @param file
 *     The file to be checked.
 * @return Returns true if the file is accepted.
 */
public boolean accept(File file)
{
    if (file.isDirectory())
    {
        return true;
    }

    String filename = file.toString().toLowerCase();

    for (int j = 0; j < imageFormats.length; j++)
    {
        if (filename.endsWith("." + imageFormats[j].toLowerCase()))
        {
            return true;
        }
    }

    return false;
}

/**
 * Returns the description.
 *
 * @return Returns the description.
 */
public String getDescription()
{
    return desc;
}
}

/**
 * Utility file filter to accept editor files, namely .xml and .xml.gz
 * extensions.
 *
 * @see ImageIO#getReaderFormatNames()
 */
public static class EditorFileFilter extends FileFilter
{
    /**
     * Description of the File format
     */
    protected String desc;

    /**
     * Constructs a new editor file filter using the specified description.
     *
     * @param description
     *     The description to use for the filter.
     */
    public EditorFileFilter(String description)
    {
        desc = description;
    }
}

```

```

/**
 * Returns true if the file is a directory or has a .xml or .xml.gz
 * extension.
 *
 * @return Returns true if the file is accepted.
 */
public boolean accept(File file)
{
    if (file.isDirectory())
    {
        return true;
    }

    String filename = file.getName().toLowerCase();

    return filename.endsWith(".xml") || filename.endsWith(".xml.gz");
}

/**
 * Returns the description.
 *
 * @return Returns the description.
 */
public String getDescription()
{
    return desc;
}
}
}

```

```
package ufsc.sisinf.brmodelo2all.ui;
```

```

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.Frame;
import java.awt.Image;
import java.awt.Toolkit;
import java.awt.event.ActionEvent;
import java.beans.PropertyVetoException;
import java.net.URL;
import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

```

```

import javax.swing.AbstractAction;
import javax.swing.Action;
import javax.swing.BorderFactory;
import javax.swing.ImageIcon;
import javax.swing.JDesktopPane;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JToolBar;
import javax.swing.SwingUtilities;
import javax.swing.UIManager;

```

```

import ufsc.sisinf.brmodelo2all.app.BrModelo2All;
import ufsc.sisinf.brmodelo2all.control.ModelingEditor;
import ufsc.sisinf.brmodelo2all.control.SqlEditor;
import ufsc.sisinf.brmodelo2all.model.Modeling;
import ufsc.sisinf.brmodelo2all.model.ModelingComponent;
import ufsc.sisinf.brmodelo2all.model.objects.AssociativeRelationObject;
import ufsc.sisinf.brmodelo2all.model.objects.AttributeObject;
import ufsc.sisinf.brmodelo2all.model.objects.ColumnObject;
import ufsc.sisinf.brmodelo2all.model.objects.InheritanceObject;
import ufsc.sisinf.brmodelo2all.model.objects.ModelingObject;
import ufsc.sisinf.brmodelo2all.model.objects.RelationObject;

import com.mxgraph.model.mxCell;
import com.mxgraph.model.mxGeometry;
import com.mxgraph.swing.util.mxGraphTransferable;
import com.mxgraph.util.mxEvent;
import com.mxgraph.util.mxEventObject;
import com.mxgraph.util.mxResources;
import com.mxgraph.util.mxEventSource.mxEventListener;

public class AppMainWindow extends JPanel {
    private static final long serialVersionUID = -6561623072112577140L;

    /**
     * Adds required resources for i18n
     */
    static {
        try {
            mxResources.add("ufsc/sisinf/brmodelo2all/ui/resources/editor");
        } catch (Exception e) {
        }
    }

    /**
     * Holds the shared number formatter.
     *
     * @see NumberFormat#getInstance()
     */
    public static final NumberFormat numberFormat = NumberFormat.getInstance();
    public static int windowCount = 0;
    public static final int EDITOR_WIDTH = 800;
    public static final int EDITOR_HEIGHT = 500;
    public static final int OBJECTS_PALETTE_WIDTH = 65;
    public static final int DIVIDER_SIZE = 3;

    public static final int CONCEPTUAL_MENU = 2;
    public static final int LOGICAL_MENU = 3;
    public static final int SELECTION_MENU = 4;
    public static final int ENTITY_PROMOTION_MENU = 2;
    public static final int ASSOCIATIVE_ENTITY_PROMOTION_MENU = 3;
    public static final int EXCLUSIVE_CONVERSION_MENU = 4;
    public static final int OPTIONAL_CONVERSION_MENU = 5;

    protected String appTitle;
    protected JPanel toolBarPanel;
    protected ToolBar toolBar;
    protected String selectedEntry = null;
    protected ModelingPalette conceptualObjects = null;
    protected ModelingPalette logicalObjects = null;
    protected JDesktopPane desktop;
    protected ModelingEditor currentEditor;
    protected List<ModelingEditor> editors;
    protected JPanel statusBar;
    protected JLabel mouseLocation;
    protected JLabel description;

```

```

public AppMainWindow() {
    this("brModeloNext");

    createModelingPalette();
    installToolBar();
}

public AppMainWindow(String appTitle) {
    // Stores and updates the frame title
    this.appTitle = appTitle;

    currentEditor = new ModelingEditor(mxResources.get("newConceptual"),
        this, true/* conceptualModeling */);
    editors = new ArrayList<ModelingEditor>();
    editors.add(currentEditor);

    desktop = new JDesktopPane();
    desktop.setBackground(Color.GRAY);
    desktop.add(currentEditor, -1);

    currentEditor.initialize();

    // Creates the status bar
    statusBar = createStatusBar();

    // Puts everything together
    setLayout(new BorderLayout());
    add(desktop, BorderLayout.CENTER);
    add(statusBar, BorderLayout.SOUTH);

    updateTitle();
}

public void createModelingPalette() {
    conceptualObjects = insertPalette(mxResources.get("objetosConceituais"));
    logicalObjects = insertPalette(mxResources.get("objetosLogicos"));

    // Sets the edge template to be used for creating new edges if an edge
    // is clicked in the shape palette
    conceptualObjects.addListener(mxEvent.SELECT, new mxEventListener() {
        public void invoke(Object sender, mxEventObject evt) {
            Object tmp = evt.getProperty("transferable");

            if (tmp instanceof mxGraphTransferable) {
                mxGraphTransferable t = (mxGraphTransferable) tmp;
                Object cell = t.getCells()[0];
                mxCell c = (mxCell) cell;
                selectedEntry = c.getValue().toString();
                if (currentEditor.getGraphComponent().getGraph().getModel()
                    .isEdge(cell))
                    ((Modeling) currentEditor.getGraphComponent()
                        .getGraph()).setEdgeTemplate(cell);
            } else
                selectedEntry = null;
        }
    });

    logicalObjects.addListener(mxEvent.SELECT, new mxEventListener() {
        public void invoke(Object sender, mxEventObject evt) {
            Object tmp = evt.getProperty("transferable");

            if (tmp instanceof mxGraphTransferable) {

```

```

        mxGraphTransferable t = (mxGraphTransferable) tmp;
        Object cell = t.getCells()[0];
        mxCell c = (mxCell) cell;
        selectedEntry = c.getValue().toString();
        if (currentEditor.getGraphComponent().getGraph().getModel()
            .isEdge(cell))
            ((Modeling) currentEditor.getGraphComponent()
                .getGraph()).setEdgeTemplate(cell);
    } else
        selectedEntry = null;
    }
});

// Adds some template cells for dropping into the graph
String text = mxResources.get("entity");
conceptualObjects
    .addTemplate(
        text,
        new ImageIcon(
            BrModelo2All.class
                .getResource("/ufsc/sisinf/brmodelo2all/ui/images/rectangle.png")),
        "entity", 300, 300, text);
text = mxResources.get("relation");
conceptualObjects
    .addTemplate(
        text,
        new ImageIcon(
            BrModelo2All.class
                .getResource("/ufsc/sisinf/brmodelo2all/ui/images/rhombus.png")),
        "relationship", 400, 180, text);
text = mxResources.get("selfRelation");
conceptualObjects
    .addTemplate(
        text,
        new ImageIcon(
            BrModelo2All.class
                .getResource("/ufsc/sisinf/brmodelo2all/ui/images/self_related.png")),
        "relationship", 400, 180, text);
text = mxResources.get("associativeEntity");
conceptualObjects
    .addTemplate(
        text,
        new ImageIcon(
            BrModelo2All.class
                .getResource("/ufsc/sisinf/brmodelo2all/ui/images/entidade_associativa.png")),
        "associativeEntity", 100, 100, text);
text = mxResources.get("inheritance");
conceptualObjects
    .addTemplate(
        text,
        new ImageIcon(
            BrModelo2All.class
                .getResource("/ufsc/sisinf/brmodelo2all/ui/images/especializacao.png")),
        "inheritance", 100, 100, text);
text = mxResources.get("exclusiveInheritance");
conceptualObjects
    .addTemplate(
        text,
        new ImageIcon(

```

```

BrModelo2All.class

.getResource("/ufsc/sisinf/brmodelo2all/ui/images/especializacao_exclusiva.png")),
    "inheritance", 100, 100, text);
text = mxResources.get("nonExclusiveInheritance");
conceptualObjects
    .addTemplate(
        text,
        new ImageIcon(
            BrModelo2All.class

.getResource("/ufsc/sisinf/brmodelo2all/ui/images/especializacao_ naoexclusiva.png")),
    "inheritance", 100, 100, text);
text = mxResources.get("attribute");
conceptualObjects
    .addTemplate(
        text,
        new ImageIcon(
            BrModelo2All.class

.getResource("/ufsc/sisinf/brmodelo2all/ui/images/atributo.png")),
    "attribute", 50, 20, text);
text = mxResources.get("identifierAttribute");
conceptualObjects
    .addTemplate(
        text,
        new ImageIcon(
            BrModelo2All.class

.getResource("/ufsc/sisinf/brmodelo2all/ui/images/atributo_identificador.png")),
    "identifierAttribute", 100, 100, text);
text = mxResources.get("composedAttribute");
conceptualObjects
    .addTemplate(
        text,
        new ImageIcon(
            BrModelo2All.class

.getResource("/ufsc/sisinf/brmodelo2all/ui/images/atributo_composto.png")),
    "attribute", 100, 100, text);
text = mxResources.get("connector");
conceptualObjects
    .addEdgeTemplate(
        text,
        new ImageIcon(
            BrModelo2All.class

.getResource("/ufsc/sisinf/brmodelo2all/ui/images/entity.png")),
    "connector", 100, 100, text);
text = mxResources.get("table");
logicalObjects
    .addTemplate(
        text,
        new ImageIcon(
            BrModelo2All.class

.getResource("/ufsc/sisinf/brmodelo2all/ui/images/tabela.png")),
    "table", 100, 100, text);
text = mxResources.get("field");
logicalObjects
    .addTemplate(
        text,
        new ImageIcon(
            BrModelo2All.class

```



```

.getResource("/ufsc/sisinf/brmodelo2all/ui/images/rectangle.png"),
                "column", 100, 100, text);
    text = mxResources.get("primaryKey");
    logicalObjects
        .addTemplate(
            text,
            new ImageIcon(
                BrModelo2All.class

.getResource("/ufsc/sisinf/brmodelo2all/ui/images/chave_primaria.png")),
                "primaryKey", 100, 100, text);
    text = mxResources.get("foreignKey");
    logicalObjects
        .addTemplate(
            text,
            new ImageIcon(
                BrModelo2All.class

.getResource("/ufsc/sisinf/brmodelo2all/ui/images/chave_estrangeira.png")),
                "foreignKey", 100, 100, text);
    text = mxResources.get("separator");
    logicalObjects
        .addTemplate(
            text,
            new ImageIcon(
                BrModelo2All.class

.getResource("/ufsc/sisinf/brmodelo2all/ui/images/separador.png")),
                "separator", 100, 100, text);
    text = mxResources.get("connector");
    logicalObjects
        .addEdgeTemplate(
            text,
            new ImageIcon(
                BrModelo2All.class

.getResource("/ufsc/sisinf/brmodelo2all/ui/images/entity.png")),
                "connector", 100, 100, text);
    updateControls();
}

public void clearModelingPalette() {
    conceptualObjects.clearSelection();
    logicalObjects.clearSelection();
}

public void updateControls() {
    boolean conceptual, logical;
    if (currentEditor == null) {
        conceptual = false;
        logical = false;
    } else if (currentEditor.isConceptualModeling()) {
        conceptual = true;
        logical = false;
    } else {
        conceptual = false;
        logical = true;
    }

    conceptualObjects.setVisible(conceptual);
    logicalObjects.setVisible(logical);

    JFrame frame = (JFrame) SwingUtilities.getWindowForComponent(this);
    if (frame != null) {

```

```

        // updates the menu
        JMenuBar menuBar = frame.getJMenuBar();
        JMenu conceptualMenu = menuBar.getMenu(CONCEPTUAL_MENU);
        JMenu logicalMenu = menuBar.getMenu(LOGICAL_MENU);
        conceptualMenu.setVisible(conceptual);
        logicalMenu.setVisible(logical);

        // updates the toolbar
        // toolbar.getComponentAtIndex(ToolBar.ne)
    }
}

protected void installToolBar() {
    toolbarPanel = new JPanel(new BorderLayout());

    JPanel objectsPanel = new JPanel(new FlowLayout());
    toolbar = new JToolBar(this, JToolBar.HORIZONTAL);
    objectsPanel.add(toolbar);

    conceptualObjects.setBorder(BorderFactory
        .createTitledBorder(mxResources.get("objetosConceituais")));
    objectsPanel.add(conceptualObjects);

    logicalObjects.setBorder(BorderFactory.createTitledBorder(mxResources
        .get("objetosLogicos")));
    objectsPanel.add(logicalObjects);

    toolbarPanel.add(objectsPanel, BorderLayout.WEST);

    add(toolbarPanel, BorderLayout.NORTH);
}

protected JPanel createStatusBar() {
    JPanel statusBar = new JPanel(new BorderLayout());
    mouseLocation = new JLabel("");
    mouseLocation.setPreferredSize(new Dimension(70, 20));
    mouseLocation.setBorder(BorderFactory.createEmptyBorder(2, 4, 2, 4));
    statusBar.add(mouseLocation, BorderLayout.WEST);

    description = new JLabel(mxResources.get("ready"));
    description.setLayout(new FlowLayout());
    description.setBorder(BorderFactory.createEmptyBorder(2, 4, 2, 4));
    description.setFont(new Font(description.getFont().getFamily(),
        Font.BOLD, 11));
    statusBar.add(description, BorderLayout.CENTER);

    return statusBar;
}

public ModelingPalette insertPalette(String title) {
    final ModelingPalette palette = new ModelingPalette(this);
    final JScrollPane scrollPane = new JScrollPane(palette);
    scrollPane

    .setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_NEVER);
    scrollPane

    .setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);

    return palette;
}

public JDesktopPane getDesktop() {
    return desktop;
}

```

```

    }

    public String getSelectedEntry() {
        ModelingPalette currentPalette = currentEditor.isConceptualModeling() ?
conceptualObjects
                : logicalObjects;

        String selected = currentPalette.getSelectedEntry() != null ? currentPalette
                .getSelectedEntry().getToolTipText() : null;
        return selected;
    }

    public void setSelectedEntry(String entry) {
        this.selectedEntry = entry;
    }

    public ModelingEditor getCurrentEditor() {
        return currentEditor;
    }

    public void setCurrentEditor(ModelingEditor editor) {
        ModelingEditor previousEditor = this.currentEditor;

        if (previousEditor != editor) {
            if (previousEditor != null) {
                try {
                    this.currentEditor.setMaximum(false);
                    this.currentEditor.setSelected(false);
                } catch (PropertyVetoException e) {
                }
            }

            this.currentEditor = editor;

            updateControls();
            updateTitle();
        }
    }

    public List<ModelingEditor> getEditors() {
        return editors;
    }

    /**
     *
     * @param msg
     */
    public void status(String location, String msg) {
        if (location != null)
            mouseLocation.setText(location);

        String descMsg;
        if (getSelectedEntry() != null) {
            descMsg = getEntryDescription();
            description.setOpaque(true);
            description.setForeground(Color.RED);
        } else {

            if (msg != null) {
                descMsg = msg;
                description.setOpaque(true);
                description.setForeground(Color.BLACK);
            } else
                descMsg = "";
        }
    }

```

```

    }

    description.setText(descMsg.toUpperCase());
}

public String getEntryDescription() {
    String description = "";

    if (selectedEntry == mxResources.get("entity"))
        description = "Clique na modelagem para inserir uma entidade"; // TODO

        // string
    else if (selectedEntry == mxResources.get("relation"))
        description = "Clique na modelagem para inserir um relacionamento"; // TODO

        // string
    else if (selectedEntry == mxResources.get("selfRelation"))
        description = "Selecione uma entidade/entidade associativa"; // TODO

        // string
    else if (selectedEntry == mxResources.get("associativeEntity"))
        description = "Clique na modelagem para inserir uma entidade associativa"; //
TODO

        // string
    else if (selectedEntry == mxResources.get("attribute"))
        || selectedEntry == mxResources.get("identifierAttribute")
        || selectedEntry == mxResources.get("composedAttribute"))
        description = "Selecione uma entidade/relacionamento/entidade associativa"; //
TODO

        // string
    else if (selectedEntry == mxResources.get("inheritance"))
        || selectedEntry == mxResources.get("exclusiveInheritance")
        || selectedEntry == mxResources.get("nonExclusiveInheritance"))
        description = "Selecione uma entidade"; // TODO string
    else if (selectedEntry == mxResources.get("connector"))
        description = "Selecione dois objetos para conectar";
    else if (selectedEntry == mxResources.get("table"))
        description = "Clique na modelagem para inserir uma tabela"; // TODO

        // string
    else if (selectedEntry == mxResources.get("field"))
        description = "Selecione uma tabela para inserir o campo"; // TODO

        // string
    else if (selectedEntry == mxResources.get("primaryKey"))
        description = "Selecione uma tabela para inserir o campo com chave primária"; //
TODO

        // string
    else if (selectedEntry == mxResources.get("foreignKey"))
        description = "Selecione uma tabela para inserir o campo com chave estrangeira";
// TODO

//
string

    else if (selectedEntry == mxResources.get("separator"))
        description = "Selecione uma tabela para inserir o separador de campos"; //
TODO

        // string

return description;

```

```

}

public void updateTitle() {
    JFrame frame = (JFrame) SwingUtilities.getWindowForComponent(this);

    if (frame != null) {
        String title = null;
        if (currentEditor != null) {
            if (currentEditor.getCurrentFile() != null) {

                title = currentEditor.getCurrentFile().getAbsolutePath();
                if (currentEditor.isModified())
                    title += "*";

            } else {
                if (currentEditor.isConceptualModeling())
                    title = mxResources.get("newConceptual");
                else
                    title = mxResources.get("newLogical");

                title += " " + windowCount;

                if (currentEditor.isModified())
                    title += "*";

            }

            currentEditor.setTitle(title);
        }

        String newTitle = title != null ? appTitle + " - " + title
            : appTitle;
        frame.setTitle(newTitle);
    }
}

public void openSqlEditor(SqlEditor sqlEditor) {
    JFrame frame = (JFrame) SwingUtilities.getWindowForComponent(this);

    sqlEditor.setModal(true);

    // Centers inside the application frame
    int x = frame.getX() + (frame.getWidth() - sqlEditor.getWidth()) / 2;
    int y = frame.getY() + (frame.getHeight() - sqlEditor.getHeight()) / 2;
    sqlEditor.setLocation(x, y);

    // Shows the modal dialog and waits
    sqlEditor.setVisible(true);
}

public void about() {
    JFrame frame = (JFrame) SwingUtilities.getWindowForComponent(this);

    if (frame != null) {
        AboutWindow about = new AboutWindow(frame);
        about.setModal(true);

        // Centers inside the application frame
        int x = frame.getX() + (frame.getWidth() - about.getWidth()) / 2;
        int y = frame.getY() + (frame.getHeight() - about.getHeight()) / 2;
        about.setLocation(x, y);

        // Shows the modal dialog and waits
        about.setVisible(true);
    }
}

```

```

}

public void properties(Object cell, ModelingComponent modelingComponent) {
    JFrame frame = (JFrame) SwingUtilities.windowForComponent(this);

    if (frame != null) {
        PropertiesWindow properties = new PropertiesWindow(frame, cell,
            modelingComponent);
        properties.setModal(true);

        // make some adjustments for positioning the window
        mxCell cellObject = (mxCell) cell;
        if (cellObject.getValue() instanceof ColumnObject
            || cellObject.getValue() instanceof AssociativeRelationObject)
            cellObject = (mxCell) cellObject.getParent();

        mxGeometry cellGeometry = cellObject.getGeometry();

        double usedWidth = cellGeometry.getX() + cellGeometry.getWidth()
            + PropertiesWindow.WINDOW_WIDTH;
        double usedHeight = cellGeometry.getY() + cellGeometry.getHeight()
            + properties.getWindowHeight();

        int x = (int) (usedWidth > modelingComponent.getWidth() ? cellGeometry
            .getX() - PropertiesWindow.WINDOW_WIDTH
            : cellGeometry.getX() + cellGeometry.getWidth() + 20;
        int y = (int) (usedHeight > modelingComponent.getHeight() ? cellGeometry
            .getY()
            + cellGeometry.getHeight()
            - properties.getWindowHeight() : cellGeometry.getY());
        x += currentEditor.getX();
        if (x < 0)
            x = 10;
        y += currentEditor.getY() + (toolBarPanel.getHeight() * 2 - 5);
        if (y < 0)
            y = 10;

        // Centers inside the application frame
        properties.setLocation(x, y);

        // Shows the modal dialog and waits
        properties.setVisible(true);
    }
}

public void exit() {
    JFrame frame = (JFrame) SwingUtilities.windowForComponent(this);

    if (frame != null) {
        frame.dispose();
    }
}

public void setLookAndFeel(String clazz) {
    JFrame frame = (JFrame) SwingUtilities.windowForComponent(this);

    if (frame != null) {
        try {
            UIManager.setLookAndFeel(clazz);
            SwingUtilities.updateComponentTreeUI(frame);

            // Needs to assign the key bindings again
            Iterator<ModelingEditor> iterator = editors.iterator();
            while (iterator.hasNext()) {
                ModelingEditor editor = iterator.next();
            }
        }
    }
}

```

```

        editor.updateKeyboardHandler();
    }

    } catch (Exception e1) {
        e1.printStackTrace();
    }
}

public JFrame createFrame(JMenuBar menuBar) {
    JFrame frame = new JFrame();
    frame.getContentPane().add(this);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setJMenuBar(menuBar);
    menuBar.getMenu(LOGICAL_MENU).setVisible(false);

    // Updates the frame title
    updateTitle();

    URL url = this.getClass().getResource(
        "/ufsc/sisinf/brmodelo2all/ui/images/icon.png");
    Image imagemTitulo = Toolkit.getDefaultToolkit().getImage(url);
    frame.setIconImage(imagemTitulo);

    frame.setExtendedState(Frame.MAXIMIZED_BOTH);

    return frame;
}

public void updateSelectionMenu(ModelingObject object) {
    JFrame frame = (JFrame) SwingUtilities.getWindowForComponent(this);
    if (frame != null) {
        JMenuBar menuBar = frame.getJMenuBar();
        JMenu selectionMenu = menuBar.getMenu(SELECTION_MENU);
        selectionMenu.setVisible(object != null);
        selectionMenu.getItem(ENTITY_PROMOTION_MENU).setVisible(
            object instanceof AttributeObject);
        selectionMenu.getItem(ASSOCIATIVE_ENTITY_PROMOTION_MENU)
            .setVisible(object instanceof RelationObject);

        boolean exclusiveConversion = false;
        boolean inheritanceObject = object instanceof InheritanceObject;
        if (inheritanceObject)
            exclusiveConversion = ((InheritanceObject) object)
                .isExclusive();

        selectionMenu.getItem(EXCLUSIVE_CONVERSION_MENU).setVisible(
            inheritanceObject && exclusiveConversion);
        selectionMenu.getItem(OPTIONAL_CONVERSION_MENU).setVisible(
            inheritanceObject && !exclusiveConversion);
    }
}

public void updateSelectionMenu(boolean visible) {
    JFrame frame = (JFrame) SwingUtilities.getWindowForComponent(this);
    if (frame != null) {
        JMenuBar menuBar = frame.getJMenuBar();
        JMenu selectionMenu = menuBar.getMenu(SELECTION_MENU);
        selectionMenu.setVisible(visible);
        selectionMenu.getItem(ENTITY_PROMOTION_MENU).setVisible(false);
        selectionMenu.getItem(ASSOCIATIVE_ENTITY_PROMOTION_MENU)
            .setVisible(false);

        selectionMenu.getItem(EXCLUSIVE_CONVERSION_MENU).setVisible(false);
    }
}

```

```

        selectionMenu.getItem(OPTIONAL_CONVERSION_MENU).setVisible(false);
    }

}

/**
 *
 * @param name
 * @param action
 * @return a new Action bound to the specified string name
 */
public Action bind(String name, final Action action) {
    return bind(name, action, null);
}

/**
 *
 * @param name
 * @param action
 * @return a new Action bound to the specified string name and icon
 */
@SuppressWarnings("serial")
public Action bind(String name, final Action action, String iconUrl) {
    return new AbstractAction(name, (iconUrl != null) ? new ImageIcon(
        BrModelo2All.class.getResource(iconUrl)) : null) {
        public void actionPerformed(ActionEvent e) {
            action.actionPerformed(new ActionEvent(getDesktop(), e.getID(),
                e.getActionCommand()));
        }
    };
}

}

package ufsc.sisinf.brmodelo2all.model.objects;

import ufsc.sisinf.brmodelo2all.util.AppConstants;

import com.mxgraph.util.mxResources;

public class AssociativeEntityObject extends ModelingObject {

    /**
     *
     */
    private static final long serialVersionUID = -3208501055540606907L;

    private boolean selfRelated = false;

    private final int NUMBER_OF_ATTRIBUTES = 1;

    private final AssociativeRelationObject relationObject;

    public AssociativeEntityObject(String name, final AssociativeRelationObject relationObject) {
        super(name);
        this.relationObject = relationObject;
    }

    public void setSelfRelated(boolean selfRelated) {
        this.selfRelated = selfRelated;
    }

    public boolean isSelfRelated() {
        return selfRelated;
    }
}

```



```

    }

    public int attributesCount() {
        return super.attributesCount() + relationObject.attributesCount() +
NUMBER_OF_ATTRIBUTES;
    }

    public void getAttributes(int types[], String names[], String values[], boolean fieldsEnabled[]) {
        super.getAttributes(types, names, values, fieldsEnabled);

        int i = super.attributesCount();

        types[i] = AppConstants.CHECK_BOX;
        names[i] = mxResources.get("selfRelated");
        fieldsEnabled[i] = false;
        values[i++] = selfRelated ? "true" : "false";

        relationObject.setIndexForComponents(super.attributesCount() +
NUMBER_OF_ATTRIBUTES);
        relationObject.getAttributes(types, names, values, fieldsEnabled);
    }

    public void setAttributes(String values[]) {
        super.setAttributes(values);

        setSelfRelated(Boolean.parseBoolean(values[2].toString()));

        relationObject.setIndexForComponents(super.attributesCount() +
NUMBER_OF_ATTRIBUTES);
        relationObject.setAttributes(values);
    }

    public int windowHeight() {
        return 300;
    }

    public String getToolTip() {
        String tip = "Tipo: Entidade associativa<br>";

        tip += super.getToolTip();
        tip += mxResources.get("selfRelated") + ": ";
        tip += selfRelated ? mxResources.get("yes") : mxResources.get("no");
        tip += relationObject.getToolTip();

        return tip;
    }

    public String getStyle() {
        return "associativeEntity";
    }
}

```

```
package ufsc.sisinf.brmodelo2all.model.objects;
```

```
import com.mxgraph.util.mxResources;
```

```
public class AssociativeRelationObject extends ModelingObject {
```

```
    /**
     *
     */
```

```
    private static final long serialVersionUID = -6136299230865369725L;
```

```
    private int index = 0;
```

```

public AssociativeRelationObject(String name) {
    super(name);
}

public int attributesCount() {
    return super.attributesCount();
}

public void setIndexForComponents(int index){
    this.index = index;
}

public int getIndexForComponents(){
    return index;
}

public void getAttributes(int types[], String names[], String values[], boolean fieldsEnabled[]) {
    super.getAttributes(types, names, values, fieldsEnabled);
}

public String getNameLabel() {
    return mxResources.get("relationName");
}

public void setAttributes(String values[]) {
    super.setAttributes(values);
}

public int windowHeight() {
    return 220;
}

public String getToolTip() {
    String tip = "<center>Relacionamento</center>";

    tip += super.getToolTip();

    return tip;
}

public String getStyle() {
    return "associativeRelation";
}
}

package ufsc.sisinf.brmodelo2all.model.objects;

import java.awt.event.ItemEvent;
import java.awt.event.ItemListener;

import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JComponent;

import ufsc.sisinf.brmodelo2all.util.AppConstants;

import com.mxgraph.model.mxCell;
import com.mxgraph.util.mxResources;

public class AttributeObject extends ModelingObject {

/**

```

```

*
*/
private static final long serialVersionUID = 5885776804640481160L;

private boolean identifier = false;

private boolean optional = false;

private boolean composed = false;

private boolean multiValued = false;

private char minimumCardinality = '1';

private char maximumCardinality = '1';

private String type = "Texto(1)";

// ao alterar o número de atributos desta classe, alterar NUMBER_OF_ATTRIBUTES
private final int NUMBER_OF_ATTRIBUTES = 7;

public AttributeObject(String name, Object parentObject, boolean identifier) {
    super(name);
    setParentObject(parentObject);
    setIdentifier(identifier);
}

public void setIdentifier(boolean identifier) {
    this.identifier = identifier;
}

public boolean isIdentifier() {
    return identifier;
}

public void setOptional(boolean optional) {
    this.optional = optional;
}

public boolean isOptional() {
    return optional;
}

public void setComposed(boolean composed) {
    this.composed = composed;
}

public boolean isComposed() {
    return composed;
}

public void setMultiValued(boolean multiValued) {
    this.multiValued = multiValued;
}

public boolean isMultiValued() {
    return multiValued;
}

public void setMinimumCardinality(char minimumCardinality) {
    this.minimumCardinality = minimumCardinality;
}

public char getMinimumCardinality() {
    return minimumCardinality;
}

```

```

}

public void setMaximumCardinality(char maximumCardinality) {
    this.maximumCardinality = maximumCardinality;
}

public char getMaximumCardinality() {
    return maximumCardinality;
}

public void setType(String type) {
    this.type = type;
}

public String getType() {
    return type;
}

public int attributesCount() {
    return super.attributesCount() + NUMBER_OF_ATTRIBUTES;
}

public void getAttributes(int types[], String names[], String values[], boolean enabled[]) {
    super.getAttributes(types, names, values, enabled);

    int i = super.attributesCount();

    types[i] = AppConstants.CHECK_BOX;
    names[i] = mxResources.get("identifier");
    enabled[i] = true;
    values[i++] = identifier ? "true" : "false";
    types[i] = AppConstants.CHECK_BOX;
    names[i] = mxResources.get("optional");
    enabled[i] = multiValued ? true : false;
    values[i++] = optional ? "true" : "false";
    types[i] = AppConstants.CHECK_BOX;
    names[i] = mxResources.get("composed");
    enabled[i] = false;
    values[i++] = composed ? "true" : "false";
    types[i] = AppConstants.CHECK_BOX;
    names[i] = mxResources.get("multiValued");
    enabled[i] = true;
    values[i++] = multiValued ? "true" : "false";
    types[i] = AppConstants.COMBO_BOX;
    names[i] = mxResources.get("minimumCardinality");
    enabled[i] = multiValued ? true : false;
    values[i++] = Character.toString(minimumCardinality);
    types[i] = AppConstants.COMBO_BOX;
    names[i] = mxResources.get("maximumCardinality");
    enabled[i] = multiValued ? true : false;
    values[i++] = Character.toString(maximumCardinality);
    types[i] = AppConstants.TEXT_FIELD;
    names[i] = mxResources.get("type");
    enabled[i] = true;
    values[i] = type;
}

public void setAttributes(String values[]) {
    super.setAttributes(values);

    setIdentifier((Boolean.parseBoolean(values[2].toString())));
    setOptional((Boolean.parseBoolean(values[3].toString())));
    setMultiValued(((Boolean.parseBoolean(values[5].toString()))));
    setMinimumCardinality(values[6].charAt(0));
    setMaximumCardinality(values[7].charAt(0));
}

```

```

        setType(values[8]);
    }

    public int windowHeight() {
        return 360;
    }

    public String getToolTip() {
        String tip = "Tipo: Atributo<br>";

        tip += super.getToolTip();
        tip += mxResources.get("identifier") + ": ";
        tip += identifier ? mxResources.get("yes") : mxResources.get("no");
        tip += "<br>";
        tip += mxResources.get("composed") + ": ";
        tip += composed ? mxResources.get("yes") : mxResources.get("no");
        tip += "<br>";
        tip += mxResources.get("multiValued") + ": ";
        tip += multiValued ? mxResources.get("yes") : mxResources.get("no");
        tip += "<br>";
        if (multiValued) {
            tip += mxResources.get("optional") + ": ";
            tip += optional ? mxResources.get("yes") : mxResources.get("no");
            tip += "<br>";
            tip += mxResources.get("minimumCardinality") + ": ";
            tip += minimumCardinality;
            tip += "<br>";
            tip += mxResources.get("maximumCardinality") + ": ";
            tip += maximumCardinality;
            tip += "<br>";
        }
        tip += mxResources.get("type") + ": ";
        tip += type;
        tip += "<br>";

        return tip;
    }

    public String[] getComboValues(String name){

        if (mxResources.get("minimumCardinality") == name) {
            String[] values = { "0", "1" };
            return values;
        } else if (mxResources.get("maximumCardinality") == name) {
            String[] values = { "n", "1", "2", "3", "4", "5", "6", "7", "8", "9", "10",
                "11", "12", "13", "14", "15", "16",
"17", "18", "19", "20" };
            return values;
        }

        return null;
    }

    public void removeReferencesFromParents() {
        ModelingObject parentObject = (ModelingObject)((mxCell)getParentObject()).getValue();
        if (parentObject != null) {
            parentObject.removeChildObject(this);

            if (parentObject instanceof AttributeObject) {
                AttributeObject parentAttribute = (AttributeObject)parentObject;

                if (parentAttribute.getChildObjects().size() == 0)
                    parentAttribute.setComposed(false);
            }
        }
    }

```

```

}

@Override
public String toString() {
    String result = getName();
    if (multiValued)
        result += "(" + minimumCardinality + "," + maximumCardinality + ")";
    return result;
}

public String getStyle() {
    return identifier ? "identifierAttribute" : "attribute";
}

public void createHandlers(JComponent[] components) {
    int componentsNumber = components.length;
    final JComboBox maximumCardinalityCombo =
(JComboBox)components[componentsNumber - 2];
    final JComboBox minimumCardinalityCombo =
(JComboBox)components[componentsNumber - 3];
    final JCheckBox multiValuedCheck = (JCheckBox)components[componentsNumber - 4];
    final JCheckBox optionalCheck = (JCheckBox)components[componentsNumber - 6];

    ItemListener listener = new ItemListener() {

        @Override
        public void itemStateChanged(ItemEvent event) {
            if (event.getSource() == multiValuedCheck) {
                boolean selected = event.getStateChange() ==
ItemEvent.SELECTED;

                optionalCheck.setEnabled(selected);
                minimumCardinalityCombo.setEnabled(selected);
                maximumCardinalityCombo.setEnabled(selected);

                if (selected)
                    maximumCardinalityCombo.setSelectedItem("n");
                else {
                    optionalCheck.setSelected(false);
                    minimumCardinalityCombo.setSelectedItem("1");
                    maximumCardinalityCombo.setSelectedItem("1");
                }
            } else if (event.getSource() == optionalCheck) {
                if (event.getStateChange() == ItemEvent.SELECTED)
                    minimumCardinalityCombo.setSelectedItem("0");
                else
                    minimumCardinalityCombo.setSelectedItem("1");
            } else if (event.getSource() == minimumCardinalityCombo) {
                if (event.getStateChange() == ItemEvent.SELECTED)
                    optionalCheck.setSelected(minimumCardinalityCombo.getSelectedItem() == "0");
            }
        }
    };

    multiValuedCheck.addItemListener(listener);
    optionalCheck.addItemListener(listener);
    minimumCardinalityCombo.addItemListener(listener);
}

public boolean usesComponentHandler(String componentName) {
    return componentName == mxResources.get("multiValued") ||

```

```

        componentName == mxResources.get("optional");
    }
}

```

```
package ufsc.sisinf.brmodelo2all.model.shapes;
```

```
import java.awt.Rectangle;
import java.awt.Shape;
import java.awt.geom.GeneralPath;
```

```
import com.mxgraph.canvas.mxGraphics2DCanvas;
import com.mxgraph.shape.mxBasicShape;
import com.mxgraph.view.mxCellState;
```

```
public class AttributeShape extends mxBasicShape {
```

```
    /**
     *
     */
```

```
    public Shape createShape(mxGraphics2DCanvas canvas, mxCellState state)
    {
```

```
        Rectangle temp = state.getRectangle();
        int x = temp.x;
        int y = temp.y;
        int w = temp.width;
        int h = temp.height;
```

```
        GeneralPath path = new GeneralPath();
```

```
        path.moveTo(x + w/2, y + h);
        path.curveTo(x - (w * 0.165), y + h, x - (w * 0.165), y, x + w/2, y);
        path.curveTo(x + w + (w * 0.165), y, x + w + (w * 0.165), y + h, x + w/2, y + h);
        path.closePath();
```

```
        return path;
```

```
    }
```

```
}
```

```
package ufsc.sisinf.brmodelo2all.model;
```

```
public enum Cardinality {
    ONE_ONE, ZERO_ONE, ONE_N, ZERO_N;
```

```
    public static String getText(Cardinality cardinality) {
```

```
        switch(cardinality) {
            case ZERO_ONE:
                return "(0,1)";
            case ZERO_N:
                return "(0,n)";
            case ONE_ONE:
                return "(1,1)";
            case ONE_N:
                return "(1,n)";
            default:
                return "";
        }
    }
```

```
}
```

```
    public static Cardinality getValue(String text) {
        Cardinality cardinality = null;
        if (text == "(0,1)")
            cardinality = ZERO_ONE;
        else if (text == "(0,n)")
```

```

        cardinality = ZERO_N;
    else if (text == "(1,1)")
        cardinality = ONE_ONE;
    else if (text == "(1,n)")
        cardinality = ONE_N;

    return cardinality;
}

public static boolean minor(Cardinality first, Cardinality second) {
    boolean result = false;

    switch(first) {
    case ONE_ONE:
        result = second != ONE_ONE;
        break;

    case ZERO_ONE:
        result = second != ONE_ONE && second != ZERO_ONE;
        break;

    case ONE_N:
        result = second == ZERO_N;

    }

    return result;
}

public static boolean major(Cardinality first, Cardinality second) {
    boolean result = false;

    switch(first) {
    case ZERO_N:
        result = second != ZERO_N;
        break;

    case ONE_N:
        result = second != ZERO_N && second != ONE_N;
        break;

    case ZERO_ONE:
        result = second == ONE_ONE;

    }

    return result;
}
}

```

```
package ufsc.sisinf.brmodelo2all.model.objects;
```

```
import ufsc.sisinf.brmodelo2all.util.AppConstants;
```

```
import com.mxgraph.model.mxCell;
```

```
import com.mxgraph.util.mxResources;
```

```
public class ColumnObject extends ModelingObject {
```

```
    private static final long serialVersionUID = -8644657476132504248L;
```

```
    private int position = 0;
```



```

private boolean primaryKey = false;

private boolean foreignKey = false;

private String type = "Texto(1)";

private String originTable = "";

private String originField = "";

private String onUpdate = "";

private String onDelete = "";

private String complement = "";

private final int NUMBER_OF_ATTRIBUTES = 9;

public ColumnObject(String name, Object table) {
    super(name);
    setParentObject(table);
    this.position = ((TableObject)((mxCell)table).getValue()).getChildObjects().size() + 1;
}

public ColumnObject(String name, Object table, boolean primaryKey, boolean foreignKey) {
    super(name);
    setParentObject(table);
    this.primaryKey = primaryKey;
    this.foreignKey = foreignKey;
    this.position = ((TableObject)((mxCell)table).getValue()).getChildObjects().size() + 1;
}

public void setPosition(int position) {
    this.position = position;
}

public int getPosition() {
    return position;
}

public void incrementPosition() {
    position++;
}

public void decrementPosition() {
    position--;
}

public void setPrimaryKey(boolean primaryKey) {
    this.primaryKey = primaryKey;
}

public boolean isPrimaryKey() {
    return primaryKey;
}

public void setForeignKey(boolean foreignKey) {
    this.foreignKey = foreignKey;
}

public boolean isForeignKey() {
    return foreignKey;
}

public void setType(String type) {

```

```

        this.type = type;
    }

    public String getType() {
        return type;
    }

    public void setOriginTable(String originTable) {
        this.originTable = originTable;
    }

    public String getOriginTable() {
        return originTable;
    }

    public void setOriginField(String originField) {
        this.originField = originField;
    }

    public String getOriginField() {
        return originField;
    }

    public void setOnUpdate(String onUpdate) {
        this.onUpdate = onUpdate;
    }

    public String getOnUpdate() {
        return onUpdate;
    }

    public void setOnDelete(String onDelete) {
        this.onDelete = onDelete;
    }

    public String getOnDelete() {
        return onDelete;
    }

    public void setComplement(String complement) {
        this.complement = complement;
    }

    public String getComplement() {
        return complement;
    }

    public int attributesCount(){
        return super.attributesCount() + NUMBER_OF_ATTRIBUTES;
    }

    public void getAttributes(int types[], String names[], String values[], boolean enabled[]) {
        super.getAttributes(types, names, values, enabled);

        int i = super.attributesCount();

        types[i] = AppConstants.TEXT_FIELD;
        names[i] = mxResources.get("index");
        enabled[i] = false;
        values[i++] = Integer.toString(position);
        types[i] = AppConstants.CHECK_BOX;
        names[i] = mxResources.get("primaryKey");
        enabled[i] = true;
        values[i++] = primaryKey ? "true" : "false";
        types[i] = AppConstants.CHECK_BOX;
    }

```

```

names[i] = mxResources.get("foreignKey");
enabled[i] = true;
values[i++] = foreignKey ? "true" : "false";
types[i] = AppConstants.TEXT_FIELD;
names[i] = mxResources.get("type");
enabled[i] = true;
values[i++] = type;
types[i] = AppConstants.TEXT_FIELD;
names[i] = mxResources.get("originTable");
enabled[i] = false;
values[i++] = originTable;
types[i] = AppConstants.TEXT_FIELD;
names[i] = mxResources.get("originField");
enabled[i] = false;
values[i++] = originField;
types[i] = AppConstants.TEXT_FIELD;
names[i] = mxResources.get("onUpdate");
enabled[i] = false;
values[i++] = onUpdate;
types[i] = AppConstants.TEXT_FIELD;
names[i] = mxResources.get("onDelete");
enabled[i] = false;
values[i++] = onDelete;
types[i] = AppConstants.TEXT_FIELD;
names[i] = mxResources.get("complement");
enabled[i] = false;
values[i] = complement;
}

public int windowHeight() {
    return super.windowHeight() + 330;
}

public void setAttributes(String values[]) {
    super.setAttributes(values);

    setPosition(Integer.parseInt(values[2]));
    setPrimaryKey((Boolean.parseBoolean(values[3].toString())));
    setForeignKey((Boolean.parseBoolean(values[4].toString())));
    setType(values[5]);
    setOriginTable(values[6]);
    setOriginField(values[7]);
    setOnUpdate(values[8]);
    setOnDelete(values[9]);
    setComplement(values[10]);
}

public String getToolTip() {
    String tip = "Tipo: Campo<br>";

    tip += super.getToolTip();
    tip += mxResources.get("index") + ": ";
    tip += position;
    tip += "<br>";
    tip += mxResources.get("primaryKey") + ": ";
    tip += primaryKey ? mxResources.get("yes") : mxResources.get("no");
    tip += "<br>";
    tip += mxResources.get("foreignKey") + ": ";
    tip += foreignKey ? mxResources.get("yes") : mxResources.get("no");
    tip += "<br>";
    tip += mxResources.get("type") + ": ";
    tip += type;
    tip += "<br>";
    tip += mxResources.get("originTable") + ": ";
    tip += originTable;
}

```

```

        tip += "<br>";
        tip += mxResources.get("originField") + ": ";
        tip += originField;
        tip += "<br>";
        tip += mxResources.get("onUpdate") + ": ";
        tip += onUpdate;
        tip += "<br>";
        tip += mxResources.get("onDelete") + ": ";
        tip += onDelete;
        tip += "<br>";
        tip += mxResources.get("complement") + ": ";
        tip += complement;
        tip += "<br>";

        return tip;
    }
    public String getStyle() {
        if (primaryKey && foreignKey)
            return "bothKeys";
        else if (primaryKey)
            return "primaryKey";
        else if (foreignKey)
            return "foreignKey";
        else
            return "column";
    }

    public void removeReferencesFromParents() {
        ModelingObject parentObject = (ModelingObject)((mxCell)getParentObject()).getValue();
        if (parentObject != null) {
            // adjust column positions for the parent table
            if (parentObject.getChildObjects().size() > position) {
                for (int i = position; i < parentObject.getChildObjects().size(); i++) {
                    ((ColumnObject)((mxCell)parentObject.getChildObjects().get(i)).getValue()).decrementPosition();
                }
            }
            parentObject.removeChildObject(this);
        }
    }
}

package ufsc.sisinf.br.modelo2all.model.shapes;

import java.awt.Rectangle;
import java.util.Map;

import com.mxgraph.canvas.mxGraphics2DCanvas;
import com.mxgraph.shape.mxLabelShape;
import com.mxgraph.util.mxConstants;
import com.mxgraph.util.mxRectangle;
import com.mxgraph.util.mxUtils;
import com.mxgraph.view.mxCellState;

public class ColumnShape extends mxLabelShape {

    public Rectangle getImageBounds(mxGraphics2DCanvas canvas, mxCellState state)
    {
        Rectangle bounds = super.getImageBounds(canvas, state);
        Map<String, Object> style = state.getStyle();
        double scale = canvas.getScale();
        int imgWidth = (int) (mxUtils.getInt(style,

```

```

        mxConstants.STYLE_IMAGE_WIDTH,
mxConstants.DEFAULT_IMAGESIZE) * scale);
        int imgHeight = (int) (mxUtils.getInt(style,
        mxConstants.STYLE_IMAGE_HEIGHT,
mxConstants.DEFAULT_IMAGESIZE) * scale);

        mxRectangle imageBounds = new mxRectangle(bounds.x, bounds.y, imgWidth,
imgHeight);

        return imageBounds.getRectangle();
    }
}

```

```
package ufsc.sisinf.brmodelo2all.ui;
```

```

import java.awt.Color;
import java.awt.Component;
import java.awt.Image;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.image.BufferedImage;
import java.awt.print.PageFormat;
import java.awt.print.Paper;
import java.awt.print.PrinterException;
import java.awt.print.PrinterJob;
import java.beans.PropertyChangeEvent;
import java.beans.PropertyChangeListener;
import java.io.File;
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.IOException;
import java.lang.reflect.Method;
import java.net.URLDecoder;
import java.net.URLEncoder;
import java.util.HashSet;
import java.util.Iterator;
import java.util.Map;

import javax.imageio.ImageIO;
import javax.swing.AbstractAction;
import javax.swing.ImageIcon;
import javax.swing.JCheckBoxMenuItem;
import javax.swing.JColorChooser;
import javax.swing.JEditorPane;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.SwingUtilities;
import javax.swing.filechooser.FileFilter;
import javax.swing.text.html.HTML;
import javax.swing.text.html.HTMLDocument;
import javax.swing.text.html.HTMLEditorKit;

import org.w3c.dom.Document;

import ufsc.sisinf.brmodelo2all.control.ConceptualConversor;
import ufsc.sisinf.brmodelo2all.control.LogicalConversor;
import ufsc.sisinf.brmodelo2all.control.ModelingEditor;
import ufsc.sisinf.brmodelo2all.control.SqlEditor;
import ufsc.sisinf.brmodelo2all.model.Modeling;
import ufsc.sisinf.brmodelo2all.model.ModelingComponent;
import ufsc.sisinf.brmodelo2all.model.objects.AssociativeEntityObject;
import ufsc.sisinf.brmodelo2all.model.objects.AssociativeRelationObject;

```

```

import ufsc.sisinf.brmodelo2all.model.objects.AttributeObject;
import ufsc.sisinf.brmodelo2all.model.objects.RelationObject;
import ufsc.sisinf.brmodelo2all.util.AppDefaultFileFilter;

import com.mxgraph.io.mxCodec;
import com.mxgraph.io.mxGdCodec;
import com.mxgraph.io.mxVdxCodec;
import com.mxgraph.io.gd.mxGdDocument;
import com.mxgraph.model.mxCell;
import com.mxgraph.model.mxGeometry;
import com.mxgraph.model.mxGraphModel;
import com.mxgraph.model.mxICell;
import com.mxgraph.model.mxIGraphModel;
import com.mxgraph.swing.mxGraphComponent;
import com.mxgraph.swing.handler.mxConnectionHandler;
import com.mxgraph.swing.view.mxCellEditor;
import com.mxgraph.util.mxCellRenderer;
import com.mxgraph.util.mxConstants;
import com.mxgraph.util.mxRectangle;
import com.mxgraph.util.mxResources;
import com.mxgraph.util.mxUtils;
import com.mxgraph.util.png.mxPNGzTXtDecoder;
import com.mxgraph.util.png.mxPngEncodeParam;
import com.mxgraph.util.png.mxPngImageEncoder;
import com.mxgraph.view.mxGraph;

/**
 *
 */
public class CommandActions
{
    /**
     *
     * @param e
     * @return Returns the window for the given action event.
     */
    public static final AppMainWindow getEditor(ActionEvent e)
    {
        if (e.getSource() instanceof Component)
        {
            Component component = (Component) e.getSource();

            while (component != null
                && !(component instanceof AppMainWindow))
            {
                component = component.getParent();
            }

            return (AppMainWindow) component;
        }

        return null;
    }

    /**
     *
     * @param e
     * @return Returns the modeling for the given action event.
     */
    public static final Modeling getModeling(ActionEvent e)
    {
        if (e.getSource() instanceof ModelingComponent) {
            ModelingComponent modelingComponent = (ModelingComponent)
e.getSource();

            return (Modeling) modelingComponent.getGraph();
        }
    }
}

```

```

    }

    if (e.getSource() instanceof Component)
    {
        Component component = (Component) e.getSource();

        while (component != null
                && !(component instanceof AppMainWindow))
        {
            component = component.getParent();
        }

        if (component instanceof AppMainWindow) {
            AppMainWindow mainWindow = (AppMainWindow) component;
            ModelingEditor modelingEditor = mainWindow.getCurrentEditor();
            if (modelingEditor != null)
                return (Modeling)
modelingEditor.getGraphComponent().getGraph();

        }
    }

    return null;
}

/**
 *
 * @param e
 * @return Returns the modeling component for the given action event.
 */
public static final ModelingComponent getModelingComponent(ActionEvent e)
{
    if (e.getSource() instanceof ModelingComponent)
        return (ModelingComponent) e.getSource();

    if (e.getSource() instanceof Component)
    {
        Component component = (Component) e.getSource();

        while (component != null
                && !(component instanceof AppMainWindow))
        {
            component = component.getParent();
        }

        if (component instanceof AppMainWindow) {
            AppMainWindow mainWindow = (AppMainWindow) component;
            ModelingEditor modelingEditor = mainWindow.getCurrentEditor();
            if (modelingEditor != null)
                return (ModelingComponent)
modelingEditor.getGraphComponent();

        }
    }

    return null;
}

/**
 *
 */
@SuppressWarnings("serial")
public static class ExitAction extends AbstractAction
{
    /**

```

```

    *
    */
    public void actionPerformed(ActionEvent e)
    {
        AppMainWindow editor = getEditor(e);

        if (editor != null)
        {
            editor.exit();
        }
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class CloseAction extends AbstractAction
{
    /**
     *
     */
    public void actionPerformed(ActionEvent e) {
        AppMainWindow editor = getEditor(e);

        if (editor != null) {
            ModelingEditor currentEditor = editor.getCurrentEditor();
            if(currentEditor != null)
                currentEditor.dispose();
        }
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class StylesheetAction extends AbstractAction
{
    /**
     *
     */
    protected String stylesheet;

    /**
     *
     */
    public StylesheetAction(String stylesheet)
    {
        this.stylesheet = stylesheet;
    }

    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        mxGraphComponent graphComponent =
(mxGraphComponent)getModelingComponent(e);
        if (graphComponent != null)
        {
            mxGraph graph = graphComponent.getGraph();
            mxCodec codec = new mxCodec();
            Document doc = mxUtils.loadDocument(CommandActions.class
                .getResource(stylesheet).toString());

```



```

        if (doc != null)
        {
            codec.decode(doc.getDocumentElement(),
                graph.getStylesheet());
            graph.refresh();
        }
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class PageSetupAction extends AbstractAction
{
    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        mxGraphComponent graphComponent =
(mxGraphComponent)getModelingComponent(e);
        if (graphComponent != null)
        {
            PrinterJob pj = PrinterJob.getPrinterJob();
            PageFormat format = pj.pageDialog(graphComponent
                .getPageFormat());

            if (format != null)
            {
                graphComponent.setPageFormat(format);
                graphComponent.zoomAndCenter();
            }
        }
    }

    /**
     *
     */
    @SuppressWarnings("serial")
    public static class PrintAction extends AbstractAction
    {
        /**
         *
         */
        public void actionPerformed(ActionEvent e)
        {
            mxGraphComponent graphComponent =
(mxGraphComponent)getModelingComponent(e);
            if (graphComponent != null)
            {
                PrinterJob pj = PrinterJob.getPrinterJob();

                if (pj.printDialog())
                {
                    PageFormat pf = graphComponent.getPageFormat();
                    Paper paper = new Paper();
                    double margin = 36;
                    paper.setImageableArea(margin, margin, paper.getWidth()
                        - margin * 2, paper.getHeight() - margin * 2);
                    pf.setPaper(paper);
                    pj.setPrintable(graphComponent, pf);
                }
            }
        }
    }
}

```

```

        try
        {
            pj.print();
        }
        catch (PrinterException e2)
        {
            System.out.println(e2);
        }
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class SaveAction extends AbstractAction
{
    /**
     *
     */
    protected boolean showDialog;

    /**
     *
     */
    protected String lastDir = null;

    /**
     *
     */
    public SaveAction(boolean showDialog)
    {
        this.showDialog = showDialog;
    }

    /**
     * Saves XML+PNG format.
     */
    protected void saveXmlPng(ModelingEditor editor, String filename,
        Color bg) throws IOException
    {
        mxGraphComponent graphComponent = editor.getGraphComponent();
        mxGraph graph = graphComponent.getGraph();

        // Creates the image for the PNG file
        BufferedImage image = mxCellRenderer.createBufferedImage(graph,
            null, 1, bg, graphComponent.isAntiAlias(), null,
            graphComponent.getCanvas());

        // Creates the URL-encoded XML data
        mxCodec codec = new mxCodec();
        String xml = URLEncoder.encode(
            mxUtils.getXml(codec.encode(graph.getModel()), "UTF-8"));
        mxPngEncodeParam param = mxPngEncodeParam
            .getDefaultEncodeParam(image);
        param.setCompressedText(new String[] { "mxGraphModel", xml });

        // Saves as a PNG file
        FileOutputStream outputStream = new FileOutputStream(new File(
            filename));

        try
        {

```

```

        mxPngImageEncoder encoder = new
mxPngImageEncoder(outputStream,
                    param);

        if (image != null)
        {
            encoder.encode(image);

            editor.setModified(false);
            editor.setCurrentFile(new File(filename));
        }
        else
        {
            JOptionPane.showMessageDialog(graphComponent,
                    mxResources.get("noImageData"));
        }
    }
    finally
    {
        outputStream.close();
    }
}

/**
 *
 */
public void actionPerformed(ActionEvent e)
{
    ModelingEditor editor = getEditor(e).getCurrentEditor();

    if (editor != null)
    {
        mxGraphComponent graphComponent = editor.getGraphComponent();
        mxGraph graph = graphComponent.getGraph();
        FileFilter selectedFilter = null;
        AppDefaultFileFilter xmlPngFilter = new AppDefaultFileFilter(".png",
                "PNG+XML " + mxResources.get("file") + " (.png)");
        FileFilter vmlFileFilter = new AppDefaultFileFilter(".html",
                "VML " + mxResources.get("file") + " (.html)");
        String filename = null;
        boolean dialogShown = false;

        if (showDialog || editor.getCurrentFile() == null)
        {
            String wd;

            if (lastDir != null)
            {
                wd = lastDir;
            }
            else if (editor.getCurrentFile() != null)
            {
                wd = editor.getCurrentFile().getParent();
            }
            else
            {
                wd = System.getProperty("user.dir");
            }

            JFileChooser fc = new JFileChooser(wd);

            // Adds the default file format
            FileFilter defaultFilter = xmlPngFilter;
            fc.addChoosableFileFilter(defaultFilter);

```

```

// Adds special vector graphics formats and HTML
fc.addChoosableFileFilter(new AppDefaultFileFilter(".mxe",
    "mxGraph Editor " + mxResources.get("file")
    + ".mxe"));
fc.addChoosableFileFilter(new AppDefaultFileFilter(".txt",
    "Graph Drawing " + mxResources.get("file")
    + ".txt"));
fc.addChoosableFileFilter(new AppDefaultFileFilter(".svg",
    "SVG " + mxResources.get("file") + "
(svg)"));

fc.addChoosableFileFilter(vmlFileFilter);
fc.addChoosableFileFilter(new AppDefaultFileFilter(".html",
    "HTML " + mxResources.get("file") + "
.html));

// Adds a filter for each supported image format
Object[] imageFormats = ImageIO.getReaderFormatNames();

// Finds all distinct extensions
HashSet<String> formats = new HashSet<String>();

for (int i = 0; i < imageFormats.length; i++)
{
    String ext =
imageFormats[i].toString().toLowerCase();
    formats.add(ext);
}

imageFormats = formats.toArray();

for (int i = 0; i < imageFormats.length; i++)
{
    String ext = imageFormats[i].toString();
    fc.addChoosableFileFilter(new
AppDefaultFileFilter("." +
    ext, ext.toUpperCase() + " "
    + mxResources.get("file") + ".(" +
ext + ")"));
}

// Adds filter that accepts all supported image formats
fc.addChoosableFileFilter(new
AppDefaultFileFilter.ImageFileFilter(
    mxResources.get("allImages")));
fc.setFileFilter(defaultFilter);
int rc = fc.showDialog(null, mxResources.get("save"));
dialogShown = true;

if (rc != JFileChooser.APPROVE_OPTION)
{
    return;
}
else
{
    lastDir = fc.getSelectedFile().getParent();
}

filename = fc.getSelectedFile().getAbsolutePath();
selectedFilter = fc.getFileFilter();

if (selectedFilter instanceof AppDefaultFileFilter)
{
    String ext = ((AppDefaultFileFilter) selectedFilter)
        .getExtension();

```

```

        if (!filename.toLowerCase().endsWith(ext))
        {
            filename += ext;
        }
    }

    if (new File(filename).exists()
        &&
JOptionPane.showConfirmDialog(graphComponent,

    mxResources.get("overwriteExistingFile") != JOptionPane.YES_OPTION)
    {
        return;
    }
    else
    {
        filename = editor.getCurrentFile().getAbsolutePath();
    }

    try
    {
        String ext = filename
            .substring(filename.lastIndexOf('.') + 1);

        if (ext.equalsIgnoreCase("svg"))
        {
            mxUtils.writeFile(mxUtils.getXml(mxCellRenderer
                .createSvgDocument(graph, null, 1,
null, null)
                .getDocumentElement()), filename);
        }
        else if (selectedFilter == vmlFileFilter)
        {
            mxUtils.writeFile(mxUtils.getXml(mxCellRenderer
                .createVmlDocument(graph, null, 1,
null, null)
                .getDocumentElement()), filename);
        }
        else if (ext.equalsIgnoreCase("html"))
        {
            mxUtils.writeFile(mxUtils.getXml(mxCellRenderer
                .createHtmlDocument(graph, null, 1,
null, null)
                .getDocumentElement()), filename);
        }
        else if (ext.equalsIgnoreCase("mxe")
            || ext.equalsIgnoreCase("xml"))
        {
            mxCodec codec = new mxCodec();
            String xml = mxUtils.getXml(codec.encode(graph
                .getModel()));

            mxUtils.writeFile(xml, filename);

            editor.setModified(false);
            editor.setCurrentFile(new File(filename));
        }
        else if (ext.equalsIgnoreCase("txt"))
        {
            String content = mxGdCodec.encode(graph
                .getString());

            mxUtils.writeFile(content, filename);
        }
    }
}

```

```

else
{
    Color bg = null;

    if ((!ext.equalsIgnoreCase("gif") && !ext
        .equalsIgnoreCase("png"))
        || JOptionPane.showConfirmDialog(
            graphComponent,
mxResources

    .get("transparentBackground")) != JOptionPane.YES_OPTION)
    {
        bg = graphComponent.getBackground();
    }

    if (selectedFilter == xmlPngFilter
        || (editor.getCurrentFile() != null
            &&
ext.equalsIgnoreCase("png") && !dialogShown))
    {
        saveXmlPng(editor, filename, bg);
    }
    else
    {
        BufferedImage image = mxCellRenderer
            .createBufferedImage(graph,
null, 1, bg,
        graphComponent.isAntiAlias(), null,
        graphComponent.getCanvas());

        if (image != null)
        {
            ImageIO.write(image, ext, new
File(filename));
        }
        else
        {
            JOptionPane.showMessageDialog(graphComponent,
mxResources.get("noImageData"));
        }
    }
}
}
}
}

catch (Throwable ex)
{
    ex.printStackTrace();
    JOptionPane.showMessageDialog(graphComponent,
        ex.toString(), mxResources.get("error"),
        JOptionPane.ERROR_MESSAGE);
}
}
}

/**
 *
 */
@SuppressWarnings("serial")
public static class ToggleDirtyAction extends AbstractAction
{
    /**

```

```

        *
        */
        public void actionPerformed(ActionEvent e)
        {
            mxGraphComponent graphComponent =
(mxGraphComponent)getModelingComponent(e);
            if (graphComponent != null)
            {
                graphComponent.showDirtyRectangle =
!graphComponent.showDirtyRectangle;
            }
        }
    }

/**
 *
 */
@SuppressWarnings("serial")
public static class ToggleConnectModeAction extends AbstractAction
{
    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        mxGraphComponent graphComponent =
(mxGraphComponent)getModelingComponent(e);
        if (graphComponent != null)
        {
            mxConnectionHandler handler = graphComponent
                .getConnectionHandler();
            handler.setHandleEnabled(!handler.isHandleEnabled());
        }
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class ToggleCreateTargetItem extends JCheckBoxMenuItem
{
    /**
     *
     */
    public ToggleCreateTargetItem(final AppMainWindow editor, String name)
    {
        super(name);
        setSelected(true);

        addActionListener(new ActionListener()
        {
            /**
             *
             */
            public void actionPerformed(ActionEvent e)
            {
                mxGraphComponent graphComponent = editor
                    .getCurrentEditor().getGraphComponent();

                if (graphComponent != null)
                {
                    mxConnectionHandler handler = graphComponent
                        .getConnectionHandler();

```

```

        handler.setCreateTarget(!handler.isCreateTarget());
        setSelected(handler.isCreateTarget());
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class PromptPropertyAction extends AbstractAction
{
    /**
     *
     */
    protected Object target;

    /**
     *
     */
    protected String fieldname, message;

    /**
     *
     */
    public PromptPropertyAction(Object target, String message)
    {
        this(target, message, message);
    }

    /**
     *
     */
    public PromptPropertyAction(Object target, String message,
        String fieldname)
    {
        this.target = target;
        this.message = message;
        this.fieldname = fieldname;
    }

    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() instanceof Component)
        {
            try
            {
                Method getter = target.getClass().getMethod(
                    "get" + fieldname);
                Object current = getter.invoke(target);

                // TODO: Support other atomic types
                if (current instanceof Integer)
                {
                    Method setter = target.getClass().getMethod(
                        "set" + fieldname, new Class[] {
int.class });

                    String value = (String) JOptionPane.showInputDialog(

```



```

message,
null, null, current);

        (Component) e.getSource(), "Value",
        JOptionPane.PLAIN_MESSAGE,

                if (value != null)
                {
                        setter.invoke(target, Integer.parseInt(value));
                }
        }
        catch (Exception ex)
        {
                ex.printStackTrace();
        }
}

// Repaints the graph component
if (e.getSource() instanceof mxGraphComponent)
{
        mxGraphComponent graphComponent = (mxGraphComponent) e
                .getSource();
        graphComponent.repaint();
}
}

/**
 *
 */
@SuppressWarnings("serial")
public static class TogglePropertyItem extends JCheckBoxMenuItem
{
        /**
         *
         */
        public TogglePropertyItem(Object target, String name, String fieldname)
        {
                this(target, name, fieldname, false);
        }

        /**
         *
         */
        public TogglePropertyItem(Object target, String name, String fieldname,
                boolean refresh)
        {
                this(target, name, fieldname, refresh, null);
        }

        /**
         *
         */
        public TogglePropertyItem(final Object target, String name,
                final String fieldname, final boolean refresh,
                ActionListener listener)
        {
                super(name);

                // Since action listeners are processed last to first we add the given
                // listener here which means it will be processed after the one below
                if (listener != null)
                {
                        addActionListener(listener);
                }
        }
}

```

```

addActionListener(new ActionListener()
{
    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        execute(target, fieldname, refresh);
    }
});

PropertyChangeListener propertyChangeListener = new
PropertyChangeListener()
{
    /**
     * (non-Javadoc)
     * @see
java.beans.PropertyChangeListener#propertyChange(java.beans.PropertyChangeEvent)
     */
    public void propertyChange(PropertyChangeEvent evt)
    {
        if (evt.getPropertyName().equalsIgnoreCase(fieldname))
        {
            update(target, fieldname);
        }
    }
};

if (target instanceof mxGraphComponent)
{
    ((mxGraphComponent) target)
        .addPropertyChangeListener(propertyChangeListener);
}
else if (target instanceof mxGraph)
{
    ((mxGraph) target)
        .addPropertyChangeListener(propertyChangeListener);
}

update(target, fieldname);
}

/**
 *
 */
public void update(Object target, String fieldname)
{
    if (target != null && fieldname != null)
    {
        try
        {
            Method getter = target.getClass().getMethod(
                "is" + fieldname);

            if (getter != null)
            {
                Object current = getter.invoke(target);

                if (current instanceof Boolean)
                {
                    setSelected(((Boolean)
current).booleanValue());
                }
            }
        }
    }
}

```

```

        }
    }
    catch (Exception e)
    {
        // ignore
    }
}

/**
 *
 */
public void execute(Object target, String fieldname, boolean refresh)
{
    if (target != null && fieldname != null)
    {
        try
        {
            Method getter = target.getClass().getMethod(
                "is" + fieldname);
            Method setter = target.getClass().getMethod(
                "set" + fieldname, new Class[] { boolean.class
});

            Object current = getter.invoke(target);

            if (current instanceof Boolean)
            {
                boolean value = !((Boolean) current).booleanValue();
                setter.invoke(target, value);
                setSelected(value);
            }

            if (refresh)
            {
                mxGraph graph = null;

                if (target instanceof mxGraph)
                {
                    graph = (mxGraph) target;
                }
                else if (target instanceof mxGraphComponent)
                {
                    graph = ((mxGraphComponent)
target).getGraph();
                }

                graph.refresh();
            }
        }
        catch (Exception e)
        {
            // ignore
        }
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class HistoryAction extends AbstractAction
{
    /**

```

```

    *
    */
protected boolean undo;

/**
 *
 */
public HistoryAction(boolean undo)
{
    this.undo = undo;
}

/**
 *
 */
public void actionPerformed(ActionEvent e)
{
    ModelingEditor editor = getEditor(e).getCurrentEditor();

    if (editor != null)
    {
        if (undo)
        {
            editor.getUndoManager().undo();
        }
        else
        {
            editor.getUndoManager().redo();
        }
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class FontStyleAction extends AbstractAction
{
    /**
     *
     */
    protected boolean bold;

    /**
     *
     */
    public FontStyleAction(boolean bold)
    {
        this.bold = bold;
    }

    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        mxGraphComponent graphComponent =
(mxGraphComponent)getModelingComponent(e);
        if (graphComponent != null)
        {
            Component editorComponent = null;

            if (graphComponent.getCellEditor() instanceof mxCellEditor)
            {

```

```

        editorComponent = ((mxCellEditor) graphComponent
            .getCellEditor()).getEditor();
    }

    if (editorComponent instanceof JEditorPane)
    {
        JEditorPane editorPane = (JEditorPane) editorComponent;
        int start = editorPane.getSelectionStart();
        int ende = editorPane.getSelectionEnd();
        String text = editorPane.getSelectedText();

        if (text == null)
        {
            text = "";
        }

        try
        {
            HTMLEditorKit editorKit = new HTMLEditorKit();
            HTMLDocument document = (HTMLDocument)
                editorPane.getDocument();
            document.remove(start, (ende - start));
            editorKit.insertHTML(document, start, ((bold) ? "<b>"
                : "<i>") + text + ((bold) ? "</b>" :
                : "</i>"),
                0, 0, (bold) ? HTML.Tag.B :
                HTML.Tag.I);
        }
        catch (Exception ex)
        {
            ex.printStackTrace();
        }

        editorPane.requestFocus();
        editorPane.select(start, ende);
    }
    else
    {
        mxIGraphModel model =
            graphComponent.getGraph().getModel();
        model.beginUpdate();
        try
        {
            graphComponent.stopEditing(false);
            graphComponent.getGraph().toggleCellStyleFlags(
                mxConstants.STYLE_FONTSTYLE,
                (bold) ? mxConstants.FONT_BOLD
                :
                mxConstants.FONT_ITALIC);
        }
        finally
        {
            model.endUpdate();
        }
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class NewConceptualModelingAction extends AbstractAction

```

```

{
    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        AppMainWindow mainWindow = getEditor(e);

        ModelingEditor newModelingEditor = new
ModelingEditor(mxResources.get("newConceptual"), mainWindow, true);
        mainWindow.setCurrentEditor(newModelingEditor);
        mainWindow.getEditors().add(newModelingEditor);
        mainWindow.getDesktop().add(newModelingEditor, -1);
        newModelingEditor.initialize();
    }
}

@SuppressWarnings("serial")
public static class NewLogicalModelingAction extends AbstractAction
{
    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        AppMainWindow mainWindow = getEditor(e);

        ModelingEditor newModelingEditor = new
ModelingEditor(mxResources.get("newLogical"), mainWindow, false);
        mainWindow.setCurrentEditor(newModelingEditor);
        mainWindow.getEditors().add(newModelingEditor);
        mainWindow.getDesktop().add(newModelingEditor, -1);
        newModelingEditor.initialize();
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class OpenAction extends AbstractAction
{
    /**
     *
     */
    protected String lastDir;

    /**
     *
     */
    protected void resetEditor(ModelingEditor editor)
    {
        editor.setModified(false);
        editor.getUndoManager().clear();
        editor.getGraphComponent().zoomAndCenter();
    }

    /**
     * Reads XML+PNG format.
     */
    protected void openXmlPng(ModelingEditor editor, File file)
        throws IOException
    {
        Map<String, String> text = mxPNGzTXtDecoder
            .decodezTXt(new FileInputStream(file));
    }
}

```

```

        if (text != null)
        {
            String value = text.get("mxGraphModel");

            if (value != null)
            {
                Document document =
mxUtils.parseXml(URLDecoder.decode(
                    value, "UTF-8"));
                mxCodec codec = new mxCodec(document);
                codec.decode(document.getDocumentElement(), editor

.getGraphComponent().getGraph().getModel());
                editor.setCurrentFile(file);
                resetEditor(editor);

                return;
            }
        }

        JOptionPane.showMessageDialog(editor,
            mxResources.get("imageContainsNoDiagramData"));
    }

    /**
     * @throws IOException
     */
    protected void openVdx(ModelingEditor editor, File file,
        Document document)
    {
        mxGraph graph = editor.getGraphComponent().getGraph();

        // Replaces file extension with .mxe
        String filename = file.getName();
        filename = filename.substring(0, filename.length() - 4) + ".mxe";

        if (new File(filename).exists()
            && JOptionPane.showConfirmDialog(editor,
                mxResources.get("overwriteExistingFile")) !=
JOptionPane.YES_OPTION)
        {
            return;
        }

        ((mxGraphModel) graph.getModel()).clear();
        mxVdxCodec.decode(document, graph);
        editor.getGraphComponent().zoomAndCenter();
        editor.setCurrentFile(new File(lastDir + "/" + filename));
    }

    /**
     * @throws IOException
     */
    protected void openGD(ModelingEditor editor, File file,
        mxGdDocument document)
    {
        mxGraph graph = editor.getGraphComponent().getGraph();

        // Replaces file extension with .mxe
        String filename = file.getName();
        filename = filename.substring(0, filename.length() - 4) + ".mxe";

```

```

        if (new File(filename).exists()
            && JOptionPane.showConfirmDialog(editor,
                mxResources.get("overwriteExistingFile")) !=
JOptionPane.YES_OPTION)
        {
            return;
        }

        ((mxGraphModel) graph.getModel()).clear();
        mxGdCodec.decode(document, graph);
        editor.getGraphComponent().zoomAndCenter();
        editor.setCurrentFile(new File(lastDir + "/" + filename));
    }

    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        ModelingEditor editor = getEditor(e).getCurrentEditor();

        if (editor != null)
        {
            if (!editor.isModified()
                || JOptionPane.showConfirmDialog(editor,
                    mxResources.get("loseChanges")) ==
JOptionPane.YES_OPTION)
            {
                mxGraph graph = editor.getGraphComponent().getGraph();

                if (graph != null)
                {
                    String wd = (lastDir != null) ? lastDir : System
                        .getProperty("user.dir");

                    JFileChooser fc = new JFileChooser(wd);

                    // Adds file filter for supported file format
                    AppDefaultFileFilter defaultFilter = new
AppDefaultFileFilter(
                        ".mxe",
                        mxResources.get("allSupportedFormats")
                            + " (.mxe, .png,
.vdx)")
                    {
                        public boolean accept(File file)
                        {
                            String lcase =
file.getName().toLowerCase();

                            return super.accept(file)
                                ||
                                ||
                                lcase.endsWith(".png")
                                ||
                                lcase.endsWith(".vdx");
                        }
                    };
                    fc.addChoosableFileFilter(defaultFilter);

                    fc.addChoosableFileFilter(new
AppDefaultFileFilter(".mxe",
                        mxResources.get("file")
                            + "mxGraph Editor " +
                        + " (.mxe)"));
                }
            }
        }
    }

```



```

AppDefaultFileFilter(".png",
mxResources.get("file")

AppDefaultFileFilter(".vdx",
mxResources.get("file")

AppDefaultFileFilter(".txt",
mxResources.get("file")

(fc.getSelectedFile().getAbsolutePath()
.toLowerCase().endsWith(".png"))
fc.getSelectedFile());
(fc.getSelectedFile().getAbsolutePath()
.toLowerCase().endsWith(".txt"))
new mxGdDocument();
document.parse(mxUtils.readFile(fc
.getSelectedFile()
.getAbsolutePath()));
fc.getSelectedFile(),
document);
}
else
{
Document document =
fc.addChoosableFileFilter(new
"PNG+XML " +
+ ".png"));
// Adds file filter for VDX import
fc.addChoosableFileFilter(new
"XML Drawing " +
+ ".vdx"));
// Adds file filter for GD import
fc.addChoosableFileFilter(new
"Graph Drawing " +
+ ".txt"));
fc.setFileFilter(defaultFilter);
int rc = fc.showDialog(null,
mxResources.get("openFile"));
if (rc == JFileChooser.APPROVE_OPTION)
{
lastDir = fc.getSelectedFile().getParent();
try
{
if
{
openXmlPng(editor,
}
else if
{
mxGdDocument document =
openGD(editor,
document);
}
else
{
Document document =

```

```

        .getAbsolutePath());

        (fc.getSelectedFile().getAbsolutePath()
            .toLowerCase().endsWith(".vdx"))

        fc.getSelectedFile(),
        document);

    new mxCodec(document);

    document.getDocumentElement(),
    graph.getModel());
    editor.setCurrentFile(fc
        .getSelectedFile());

        }
        resetEditor(editor);
    }
    catch (IOException ex)
    {
        ex.printStackTrace();
        JOptionPane.showMessageDialog(

            editor.getGraphComponent(),
                ex.toString(),

            mxResources.get("error"),
            JOptionPane.ERROR_MESSAGE);
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class ToggleAction extends AbstractAction
{
    /**
     *
     */
    protected String key;

    /**
     *
     */
    protected boolean defaultValue;

```

```

/**
 *
 * @param key
 */
public ToggleAction(String key)
{
    this(key, false);
}

/**
 *
 * @param key
 */
public ToggleAction(String key, boolean defaultValue)
{
    this.key = key;
    this.defaultValue = defaultValue;
}

/**
 *
 */
public void actionPerformed(ActionEvent e)
{
    mxGraph graph = getModeling(e);

    if (graph != null)
    {
        graph.toggleCellStyles(key, defaultValue);
    }
}

}

/**
 *
 */
@SuppressWarnings("serial")
public static class SetLabelPositionAction extends AbstractAction
{
    /**
     *
     */
    protected String labelPosition, alignment;

    /**
     *
     * @param key
     */
    public SetLabelPositionAction(String labelPosition, String alignment)
    {
        this.labelPosition = labelPosition;
        this.alignment = alignment;
    }

    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        mxGraph graph = getModeling(e);

        if (graph != null && !graph.isSelectionEmpty())
        {
            graph.getModel().beginUpdate();

```

```

        try
        {
            // Checks the orientation of the alignment to use the correct
constants
            if (labelPosition.equals(mxConstants.ALIGN_LEFT)
                ||
labelPosition.equals(mxConstants.ALIGN_CENTER)
                ||
labelPosition.equals(mxConstants.ALIGN_RIGHT))
            {
                graph.setCellStyles(mxConstants.STYLE_LABEL_POSITION,
                                    labelPosition);
                graph.setCellStyles(mxConstants.STYLE_ALIGN,
alignment);
            }
            else
            {
                graph.setCellStyles(
                    mxConstants.STYLE_VERTICAL_LABEL_POSITION,
                                    labelPosition);
                graph.setCellStyles(mxConstants.STYLE_VERTICAL_ALIGN,
                                    alignment);
            }
        }
        finally
        {
            graph.getModel().endUpdate();
        }
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class SetStyleAction extends AbstractAction
{
    /**
     *
     */
    protected String value;

    /**
     *
     * @param key
     */
    public SetStyleAction(String value)
    {
        this.value = value;
    }

    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        mxGraph graph = getModeling(e);

        if (graph != null && !graph.isSelectionEmpty())
        {
            graph.setCellStyle(value);
        }
    }
}

```

```

    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class KeyValueAction extends AbstractAction
{
    /**
     *
     */
    protected String key, value;

    /**
     *
     * @param key
     */
    public KeyValueAction(String key)
    {
        this(key, null);
    }

    /**
     *
     * @param key
     */
    public KeyValueAction(String key, String value)
    {
        this.key = key;
        this.value = value;
    }

    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        mxGraph graph = getModeling(e);

        if (graph != null && !graph.isSelectionEmpty())
        {
            graph.setCellStyles(key, value);
        }
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class PromptValueAction extends AbstractAction
{
    /**
     *
     */
    protected String key, message;

    /**
     *
     * @param key
     */
    public PromptValueAction(String key, String message)
    {

```

```

        this.key = key;
        this.message = message;
    }

    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        if (e.getSource() instanceof Component)
        {
            mxGraph graph = getModeling(e);

            if (graph != null && !graph.isSelectionEmpty())
            {
                String value = (String) JOptionPane.showInputDialog(
                    (Component) e.getSource(),
                    mxResources.get("value"), message,
                    JOptionPane.PLAIN_MESSAGE, null, null,
                    "");

                if (value != null)
                {
                    if (value.equals(mxConstants.NONE))
                    {
                        value = null;
                    }

                    graph.setCellStyles(key, value);
                }
            }
        }
    }

    /**
     *
     */
    @SuppressWarnings("serial")
    public static class AlignCellsAction extends AbstractAction
    {
        /**
         *
         */
        protected String align;

        /**
         *
         * @param key
         */
        public AlignCellsAction(String align)
        {
            this.align = align;
        }

        /**
         *
         */
        public void actionPerformed(ActionEvent e)
        {
            mxGraph graph = getModeling(e);

            if (graph != null && !graph.isSelectionEmpty())
            {
                graph.alignCells(align);
            }
        }
    }
}

```

```

    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class AutosizeAction extends AbstractAction
{
    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        mxGraph graph = getModeling(e);

        if (graph != null && !graph.isSelectionEmpty())
        {
            Object[] cells = graph.getSelectionCells();
            mxIGraphModel model = graph.getModel();

            model.beginUpdate();
            try
            {
                for (int i = 0; i < cells.length; i++)
                {
                    graph.updateCellSize(cells[i]);
                }
            }
            finally
            {
                model.endUpdate();
            }
        }
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class ColorAction extends AbstractAction
{
    /**
     *
     */
    protected String name, key;

    /**
     *
     * @param key
     */
    public ColorAction(String name, String key)
    {
        this.name = name;
        this.key = key;
    }

    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {

```

```

        mxGraphComponent graphComponent =
(mxGraphComponent)getModelingComponent(e);
        if (graphComponent != null)
        {
            mxGraph graph = graphComponent.getGraph();

            if (!graph.isSelectionEmpty())
            {
                Color newColor =
JColorChooser.showDialog(graphComponent,
                            name, null);

                if (newColor != null)
                {
                    graph.setCellStyles(key,
mxUtils.hexString(newColor));
                }
            }
        }
    }

/**
 *
 */
@SuppressWarnings("serial")
public static class BackgroundImageAction extends AbstractAction
{
    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        mxGraphComponent graphComponent =
(mxGraphComponent)getModelingComponent(e);
        if (graphComponent != null)
        {
            String value = (String) JOptionPane.showInputDialog(
mxResources.get("backgroundImage"),
                            "URL", JOptionPane.PLAIN_MESSAGE, null, null,
                            "http://www.callatecs.com/images/background2.JPG");

            if (value != null)
            {
                if (value.length() == 0)
                {
                    graphComponent.setBackgroundImage(null);
                }
                else
                {
                    Image background = mxUtils.loadImage(value);
                    // Incorrect URLs will result in no image.
                    // TODO provide feedback that the URL is not correct
                    if (background != null)
                    {
                        graphComponent.setBackgroundImage(new
ImageIcon(
                            background));
                    }
                }

                // Forces a repaint of the outline
                graphComponent.getGraph().repaint();
            }
        }
    }
}

```



```

    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class BackgroundAction extends AbstractAction
{
    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        mxGraphComponent graphComponent =
(mxGraphComponent)getModelingComponent(e);
        if (graphComponent != null)
        {
            Color newColor = JColorChooser.showDialog(graphComponent,
                mxResources.get("background"), null);

            if (newColor != null)
            {
                graphComponent.getViewport().setOpaque(false);
                graphComponent.setBackground(newColor);
            }

            // Forces a repaint of the outline
            graphComponent.getGraph().repaint();
        }
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class PageBackgroundAction extends AbstractAction
{
    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        mxGraphComponent graphComponent =
(mxGraphComponent)getModelingComponent(e);
        if (graphComponent != null)
        {
            Color newColor = JColorChooser.showDialog(graphComponent,
                mxResources.get("pageBackground"), null);

            if (newColor != null)
            {
                graphComponent.setPageBackgroundColor(newColor);
            }

            // Forces a repaint of the component
            graphComponent.repaint();
        }
    }
}

/**
 *
 */

```

```

    */
    @SuppressWarnings("serial")
    public static class StyleAction extends AbstractAction
    {
        /**
         *
         */
        public void actionPerformed(ActionEvent e)
        {
            mxGraphComponent graphComponent =
(mxGraphComponent)getModelingComponent(e);
            if (graphComponent != null)
            {
                mxGraph graph = graphComponent.getGraph();
                String initial = graph.getModel().getStyle(
                    graph.getSelectionCell());
                String value = (String) JOptionPane.showInputDialog(
                    graphComponent, mxResources.get("style"),
                    mxResources.get("style"),
                    JOptionPane.PLAIN_MESSAGE,
                    null, null, initial);

                if (value != null)
                {
                    graph.setCellStyle(value);
                }
            }
        }
    }

    /**
     *
     */
    @SuppressWarnings("serial")
    public static class InsertConceptualObjectAction extends AbstractAction
    {
        /**
         *
         */
        protected String name;

        /**
         *
         * @param key
         */
        public InsertConceptualObjectAction(String name)
        {
            this.name = name;
        }

        /**
         *
         */
        public void actionPerformed(ActionEvent e)
        {
            AppMainWindow mainWindow = getEditor(e);

            // clear selection first, so it won't select and unselect the object through the menu
            mainWindow.conceptualObjects.clearSelection();

            int count = mainWindow.conceptualObjects.getComponentCount();
            int index = 0;
            boolean found = false;
            while(!found && index < count) {

```

```

        Component component =
mainWindow.conceptualObjects.getComponent(index);
        if (component instanceof JLabel) {
            JLabel label = (JLabel)component;
            if (label.getToolTipText() == name) {

mainWindow.conceptualObjects.setSelectionEntry(label,

mainWindow.conceptualObjects.getTransferable(index));
                found = true;
            }
        }
        index++;
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class InsertLogicalObjectAction extends AbstractAction
{
    /**
     *
     */
    protected String name;

    /**
     *
     * @param key
     */
    public InsertLogicalObjectAction(String name)
    {
        this.name = name;
    }

    /**
     *
     */
    public void actionPerformed(ActionEvent e)
    {
        AppMainWindow mainWindow = getEditor(e);

        // clear selection first, so it won't select and unselect the object through the menu
bar
        mainWindow.logicalObjects.clearSelection();

        int count = mainWindow.logicalObjects.getComponentCount();
        int index = 0;
        boolean found = false;
        while(!found && index < count) {
            Component component =
mainWindow.logicalObjects.getComponent(index);
            if (component instanceof JLabel) {
                JLabel label = (JLabel)component;
                if (label.getToolTipText() == name) {
                    mainWindow.logicalObjects.setSelectionEntry(label,

mainWindow.logicalObjects.getTransferable(index));
                        found = true;
                    }
                }
            index++;
        }
}

```

```

    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class ConvertConceptualToLogicalAction extends AbstractAction {
    /**
     *
     */
    public void actionPerformed(ActionEvent e) {
        ModelingComponent modelingComponent = getModelingComponent(e);
        AppMainWindow mainWindow = getEditor(e);

        ModelingEditor newModelingEditor = new
ModelingEditor(mxResources.get("newLogical"), mainWindow, false);
mainWindow.setCurrentEditor(newModelingEditor);
mainWindow.getEditors().add(newModelingEditor);
mainWindow.getDesktop().add(newModelingEditor, -1);
newModelingEditor.initialize();

        ConceptualConversor conversor = new
ConceptualConversor(modelingComponent, newModelingEditor, mainWindow);
conversor.convertModeling();
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class ConvertLogicalToPhysicalAction extends AbstractAction {
    /**
     *
     */
    public void actionPerformed(ActionEvent e) {
        ModelingComponent modelingComponent = getModelingComponent(e);
        AppMainWindow mainWindow = getEditor(e);
        JFrame frame = (JFrame) SwingUtilities.windowForComponent(mainWindow);
        SqlEditor sqlEditor = new SqlEditor(frame);

        LogicalConversor conversor = new LogicalConversor(modelingComponent,
sqlEditor);
conversor.convertModeling();

        mainWindow.openSqlEditor(sqlEditor);
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class AssociativeEntityPromotionAction extends AbstractAction {
    /**
     *
     */
    public void actionPerformed(ActionEvent e) {
        mxGraphComponent graphComponent =
(mxGraphComponent)getModelingComponent(e);
        if (graphComponent != null) {
            mxGraph graph = graphComponent.getGraph();

            if (!graph.isSelectionEmpty()) {

```

```

        promote(graph);
    }
}

public void promote(mxGraph graph) {
    Object selectedCell = graph.getSelectionCell();
    graph.getModel().beginUpdate();
    try {
        mxCell relationCell = (mxCell)selectedCell;
        RelationObject relation = (RelationObject)relationCell.getValue();
        mxGeometry geometry = relationCell.getGeometry();
        mxRectangle newBounds = new mxRectangle(geometry.getX(),
geometry.getY(), 150, 80);

        graph.getModel().setStyle(relationCell, "associativeEntity");
        graph.resizeCell(relationCell, newBounds);

        String relationName = relation.getName();
        AssociativeRelationObject associativeRelationObject = new
AssociativeRelationObject(relationName); //NON-NLS-1$
        AssociativeEntityObject entityObject = new
AssociativeEntityObject(mxResources.get("associativeEntity"), associativeRelationObject); //NON-NLS-
1$

        relationCell.setValue(entityObject);
        Object newRelationObject = graph.insertVertex(relationCell, null,
associativeRelationObject, 15, 20, 120, 50, "associativeRelation");

        entityObject.addChildObject(newRelationObject); //NON-NLS-1$
        Iterator<Object> iterator = relation.getChildObjects().iterator();
        while (iterator.hasNext()) {
            Object object = iterator.next();

            associativeRelationObject.addChildObject(object);

            ((AttributeObject)((mxCell)object).getValue()).setParentObject(newRelationObject);
        }
        relation.getChildObjects().clear();

        int count = relationCell.getEdgeCount();
        for (int i = 0; i < count; i++) {
            mxCell edge = (mxCell) relationCell.getEdgeAt(i);
            if (edge.getSource() == relationCell)
                edge.setSource((mxICell) newRelationObject);
            else
                edge.setTarget((mxICell) newRelationObject);
        }
    } finally {
        graph.getModel().endUpdate();
    }
}

/**
 *
 */
@SuppressWarnings("serial")
public static class EntityPromotionAction extends AbstractAction {
    /**
     *
     */
    public void actionPerformed(ActionEvent e) {

```



```

@Override
public String toString() {
    return Cardinality.getText(cardinality);
}

public void setCardinality(Cardinality cardinality) {
    this.cardinality = cardinality;
}

public Cardinality getCardinality() {
    return cardinality;
}

public void setWeakEntity(boolean weakEntity) {
    this.weakEntity = weakEntity;
}

public boolean isWeakEntity() {
    return weakEntity;
}

public int attributesCount() {
    return super.attributesCount() + NUMBER_OF_ATTRIBUTES;
}

public void getAttributes(int types[], String names[], String values[], boolean enabled[]) {
    super.getAttributes(types, names, values, enabled);

    int i = super.attributesCount();

    types[i] = AppConstants.CHECK_BOX;
    names[i] = mxResources.get("weakEntity");
    enabled[i] = true;
    values[i++] = weakEntity ? "true" : "false";
    types[i] = AppConstants.COMBO_BOX;
    names[i] = mxResources.get("cardinality");
    enabled[i] = true;
    values[i] = Cardinality.getText(cardinality);
}

public void setAttributes(String values[]) {
    super.setAttributes(values);

    setWeakEntity(Boolean.parseBoolean(values[2].toString()));
    setCardinality(Cardinality.getValue(values[3].toString()));
}

public int windowHeight() {
    return 220;
}

public String getToolTip() {
    String tip = "Tipo: Cardinalidade<br>";

    tip += super.getToolTip();
    tip += mxResources.get("weakEntity") + ": ";
    tip += weakEntity ? mxResources.get("yes") : mxResources.get("no");
    tip += "<br>";
    tip += mxResources.get("cardinality") + ": ";
    tip += Cardinality.getText(cardinality);
    tip += "<br>";

    return tip;
}

```

```

public String[] getComboValues(String name){
    if (mxResources.get("cardinality") == name) {
        String[] values = { "(0,1)", "(0,n)", "(1,1)", "(1,n)" };
        return values;
    }
    return null;
}

public String getStyle() {
    return weakEntity ? "weakEntity" : "entityRelationConnector";
}

public String getNameLabel() {
    return mxResources.get("paper");
}
}

```

```
package ufsc.sisinf.br.modelo2all.ui;
```

```

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.Frame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRadioButton;

```

```
public class ConversionIteratorWindow extends JDialog {
```

```

    /**
     *
     */
    private static final long serialVersionUID = -2955881058927788521L;

    private int result = -1;

    public ConversionIteratorWindow(Frame arg0, String[] messages, String[] alternatives, int
suggestionIndex) {
        super(arg0);

        setLayout(new BorderLayout());

        JPanel messagePanel = createMessagePanel(messages);
        getContentPane().add(messagePanel, BorderLayout.NORTH);

        JPanel questionsPanel = createQuestionsPanel(alternatives, suggestionIndex);
        getContentPane().add(questionsPanel, BorderLayout.CENTER);

        // panel dos botões
        JPanel buttonsPanel = createButtonsPanel();
        getContentPane().add(buttonsPanel, BorderLayout.SOUTH);

        setResizable(false);
        setSize(500, 215);
    }
}

```



```

}

public JPanel createMessagePanel2() {
    JPanel p = new JPanel();
    p.setLayout(new BorderLayout(p, BorderLayout.Y_AXIS));
    p.setBorder(BorderFactory.createEmptyBorder(2, 10, 2, 5));
    JLabel text = new JLabel("O que fazer a respeito do relacionamento (0,1)<->(0,n)");
    text.setFont(new Font(text.getFont().getFamily(), Font.BOLD, 12));
    text.setBorder(BorderFactory.createEmptyBorder(5, 0, 0, 0));
    p.add(text);
    text = new JLabel("encontrado entre as entidades \"Pessoas\" e \"Automóveis\"?");
    text.setFont(new Font(text.getFont().getFamily(), Font.BOLD, 12));
    text.setBorder(BorderFactory.createEmptyBorder(5, 0, 10, 0));
    p.add(text);

    return p;
}

public JPanel createMessagePanel(String[] messages) {
    JPanel p = new JPanel();
    p.setLayout(new BorderLayout(p, BorderLayout.Y_AXIS));
    p.setBorder(BorderFactory.createEmptyBorder(2, 10, 2, 5));

    for (int i = 0; i < messages.length; i++) {
        JLabel text = new JLabel(messages[i]);
        text.setFont(new Font(text.getFont().getFamily(), Font.BOLD, 12));
        text.setBorder(BorderFactory.createEmptyBorder(5, 0, 0, 0));
        p.add(text);
    }

    return p;
}

public JPanel createQuestionsPanel2() {
    JPanel p = new JPanel();
    p.setLayout(new BorderLayout(p, BorderLayout.Y_AXIS));
    p.setBorder(BorderFactory.createEmptyBorder(0, 5, 0, 5));

    JRadioButton firstOption = new JRadioButton("1) Adicionar a(s) chave(s) da tabela de
menor cardinalidade à tabela de maior cardinalidade", true);
    JRadioButton secondOption = new JRadioButton("2) Criar uma tabela para o
relacionamento", false);
    JRadioButton thirdOption = new JRadioButton("3) Deste ponte em diante aceitar todas as
sugestões", false);
    p.add(firstOption);
    p.add(secondOption);
    p.add(thirdOption);

    return p;
}

public JPanel createQuestionsPanel(String[] alternatives, int suggestionIndex) {
    JPanel p = new JPanel();
    p.setLayout(new BorderLayout(p, BorderLayout.Y_AXIS));
    p.setBorder(BorderFactory.createEmptyBorder(0, 5, 0, 5));

    ButtonGroup radioGroup = new ButtonGroup();
    for (int i = 0; i < alternatives.length; i++) {
        JRadioButton alternative = new JRadioButton(alternatives[i], i ==
suggestionIndex);
        p.add(alternative);
        radioGroup.add(alternative);
    }

    return p;
}

```

```

    }

    public JPanel createButtonsPanel() {
        JPanel p = new JPanel(new FlowLayout(FlowLayout.CENTER));
        p.setBorder(BorderFactory.createEmptyBorder(8, 0, 0, 0));
        // Adds OK button to close window
        JButton okButton = new JButton("Ok");
        okButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                JPanel radioPanel = (JPanel) getContentPane().getComponent(1);
                for (int i = 0; i < radioPanel.getComponentCount() && result == -1;
i++) {
                    if (((JRadioButton) radioPanel.getComponent(i)).isSelected())
                        result = i;
                }
                setVisible(false);
            }
        });

        // Adds OK button to close window
        JButton cancelButton = new JButton("Cancelar");
        cancelButton.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                setVisible(false);
            }
        });

        p.add(okButton);
        p.add(cancelButton);

        // Sets default button for enter key
        getRootPane().setDefaultButton(okButton);

        return p;
    }

    public int getResult() {
        return result;
    }
}

```

```

package ufsc.sisinf.brmodelo2all.model.objects;

```

```

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

```

```

import ufsc.sisinf.brmodelo2all.util.AppConstants;

```

```

import com.mxgraph.model.mxCell;
import com.mxgraph.util.mxResources;

```

```

public class EntityObject extends ModelingObject {

```

```

    /**
     *
     */

```

```

    private static final long serialVersionUID = -7141376806822311393L;

```

```

    private boolean selfRelated = false;

```

```

    private boolean specialized = false;

```

```

    private final int NUMBER_OF_ATTRIBUTES = 2;

```

```

public EntityObject(String name) {
    super(name);
}

public void setSelfRelated(boolean selfRelated) {
    this.selfRelated = selfRelated;
}

public boolean isSelfRelated() {
    return selfRelated;
}

public void setSpecialized(boolean specialized) {
    this.specialized = specialized;
}

public boolean isSpecialized() {
    return specialized;
}

public int attributesCount() {
    return super.attributesCount() + NUMBER_OF_ATTRIBUTES;
}

public void getAttributes(int types[], String names[], String values[], boolean fieldsEnabled[]) {
    super.getAttributes(types, names, values, fieldsEnabled);

    int i = super.attributesCount();

    types[i] = AppConstants.CHECK_BOX;
    names[i] = mxResources.get("selfRelated");
    fieldsEnabled[i] = false;
    values[i++] = selfRelated ? "true" : "false";
    types[i] = AppConstants.CHECK_BOX;
    names[i] = mxResources.get("specialized");
    fieldsEnabled[i] = false;
    values[i] = specialized ? "true" : "false";
}

public void setAttributes(String values[]) {
    super.setAttributes(values);

    setSelfRelated(Boolean.parseBoolean(values[2].toString()));
    setSpecialized(Boolean.parseBoolean(values[3].toString()));
}

public int windowHeight() {
    return 220;
}

public String getToolTip() {
    String tip = "Tipo: Entidade <br>";

    tip += super.getToolTip();
    tip += mxResources.get("selfRelated") + ": ";
    tip += selfRelated ? mxResources.get("yes") : mxResources.get("no");
    tip += "<br>";
    tip += mxResources.get("specialized") + ": ";
    tip += specialized ? mxResources.get("yes") : mxResources.get("no");
    tip += "<br>";

    return tip;
}

```

```

public void removeReferencesFromParents() {
    mxCell parent = (mxCell)getParentObject();
    if(parent != null && parent.getValue() instanceof InheritanceObject) {
        InheritanceObject inheritance = (InheritanceObject)parent.getValue();
        if (inheritance.getChildObjects().size() == 2)
            inheritance.setExclusive(false);

        inheritance.removeChildObject(this);

        setParentObject(null);
    }
}

public String getStyle() {
    return "entity";
}

public List<AttributeObject> getPrimaryKeys() {
    List<AttributeObject> primaryKeys = new ArrayList<AttributeObject>();

    Iterator<Object> iterator = getChildObjects().iterator();
    while (iterator.hasNext()) {
        Object object = iterator.next();
        if (((mxCell)object).getValue() instanceof AttributeObject) {
            AttributeObject attribute = (AttributeObject)
                ((mxCell)object).getValue();
            if (attribute.isIdentifier())
                primaryKeys.add(attribute);
        }
    }
    return primaryKeys;
}
}

package ufsc.sisinf.brmodelo2all.model.objects;

import ufsc.sisinf.brmodelo2all.util.AppConstants;

import com.mxgraph.model.mxCell;
import com.mxgraph.util.mxResources;

public class InheritanceObject extends ModelingObject {

    /**
     *
     */
    private static final long serialVersionUID = -4514813052165457729L;

    private boolean exclusive = false;

    private boolean partial = false;

    private final int NUMBER_OF_OBJECTS = 2;

    public InheritanceObject(String name, Object parentObject) {
        super(name);
        setParentObject(parentObject);
    }

    @Override
    public String toString() {
        return partial ? "p" : "";
    }
}

```

```

public void setExclusive(boolean exclusive) {
    this.exclusive = exclusive;
}

public boolean isExclusive() {
    return exclusive;
}

public void setpartial(boolean partial) {
    this.partial = partial;
}

public boolean ispartial() {
    return partial;
}

public int attributesCount() {
    return super.attributesCount() + NUMBER_OF_OBJECTS;
}

public void getAttributes(int types[], String names[], String values[], boolean enabled[]) {
    super.getAttributes(types, names, values, enabled);

    int i = super.attributesCount();

    types[i] = AppConstants.CHECK_BOX;
    names[i] = mxResources.get("exclusive");
    enabled[i] = false;
    values[i++] = exclusive ? "true" : "false";
    types[i] = AppConstants.CHECK_BOX;
    names[i] = mxResources.get("partial");
    enabled[i] = true;
    values[i] = partial ? "true" : "false";
}

public void setAttributes(String values[]) {
    super.setAttributes(values);

    setExclusive((Boolean.parseBoolean(values[2].toString())));
    setpartial((Boolean.parseBoolean(values[3].toString())));
}

public int windowHeight() {
    return 220;
}

public String getToolTip() {
    String tip = "Tipo: Especialização <br>";

    tip += super.getToolTip();
    tip += mxResources.get("exclusive") + ": ";
    tip += exclusive ? mxResources.get("yes") : mxResources.get("no");
    tip += "<br>";
    tip += mxResources.get("partial") + ": ";
    tip += partial ? mxResources.get("yes") : mxResources.get("no");
    tip += "<br>";

    return tip;
}

public void removeReferencesFromParents() {
    Object parent = getParentObject();
    if (parent != null) {
        EntityObject entityObject = (EntityObject) ((mxCell)parent).getValue();
        entityObject.removeChildObject(this);
    }
}

```

```

        entityObject.setSpecialized(false);
    }
}

public String getStyle() {
    return partial ? "partialInheritance" : "inheritance";
}
}

package ufsc.sisinf.brmodelo2all.model.shapes;

import java.awt.Rectangle;
import java.awt.Shape;
import java.awt.geom.GeneralPath;

import com.mxgraph.canvas.mxGraphics2DCanvas;
import com.mxgraph.shape.mxBasicShape;
import com.mxgraph.view.mxCellState;

public class InheritanceShape extends mxBasicShape {

    /**
     *
     */
    public Shape createShape(mxGraphics2DCanvas canvas, mxCellState state)
    {
        Rectangle temp = state.getRectangle();
        int x = temp.x;
        int y = temp.y;
        int w = temp.width;
        int h = temp.height;

        GeneralPath path = new GeneralPath();

        path.moveTo(x + (w/2), y);
        path.lineTo(x + w , y + h);
        path.lineTo(x, y + h);
        path.lineTo(x + (w/2), y);
        path.closePath();

        return path;
    }
}

package ufsc.sisinf.brmodelo2all.ui;

import javax.swing.ActionMap;
import javax.swing.InputMap;
import javax.swing.JComponent;
import javax.swing.KeyStroke;

import com.mxgraph.swing.mxGraphComponent;
import com.mxgraph.swing.handler.mxKeyboardHandler;
import com.mxgraph.swing.util.mxGraphActions;

/**
 * @author Administrator
 *
 */
public class KeyboardHandler extends mxKeyboardHandler
{
    /**

```

```

*
* @param graphComponent
*/
public KeyboardHandler(mxGraphComponent graphComponent)
{
    super(graphComponent);
}

/**
 * Return JTree's input map.
 */
protected InputMap getInputMap(int condition)
{
    InputMap map = super.getInputMap(condition);

    if (condition == JComponent.WHEN_FOCUSED && map != null)
    {
        map.put(KeyStroke.getKeyStroke("control S"), "save");
        map.put(KeyStroke.getKeyStroke("control shift S"), "saveAs");
        map.put(KeyStroke.getKeyStroke("control N"), "new");
        map.put(KeyStroke.getKeyStroke("control O"), "open");

        map.put(KeyStroke.getKeyStroke("control Z"), "undo");
        map.put(KeyStroke.getKeyStroke("control Y"), "redo");
        map
            .put(KeyStroke.getKeyStroke("control shift V"),
                "selectVertices");
        map.put(KeyStroke.getKeyStroke("control shift E"), "selectEdges");
    }

    return map;
}

/**
 * Return the mapping between JTree's input map and JGraph's actions.
 */
protected ActionMap createActionMap()
{
    ActionMap map = super.createActionMap();

    map.put("save", new CommandActions.SaveAction(false));
    map.put("saveAs", new CommandActions.SaveAction(true));
    map.put("newConceptual", new CommandActions.NewConceptualModelingAction());
    map.put("newLogical", new CommandActions.NewLogicalModelingAction());
    map.put("open", new CommandActions.OpenAction());
    map.put("close", new CommandActions.CloseAction());
    map.put("undo", new CommandActions.HistoryAction(true));
    map.put("redo", new CommandActions.HistoryAction(false));
    map.put("selectVertices", mxGraphActions.getSelectVerticesAction());
    map.put("selectEdges", mxGraphActions.getSelectEdgesAction());

    return map;
}
}

package ufsc.sisinf.br.modelo2all.ui;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.AbstractAction;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;

```

```

import javax.swing.TransferHandler;
import javax.swing.UIManager;

import ufsc.sisinf.brmodelo2all.ui.CommandActions.AlignCellsAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.AssociativeEntityPromotionAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.AutosizeAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.CloseAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.ColorAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.ConvertConceptualToLogicalAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.ConvertLogicalToPhysicalAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.EntityPromotionAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.ExitAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.HistoryAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.InsertConceptualObjectAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.InsertLogicalObjectAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.NewConceptualModelingAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.NewLogicalModelingAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.OpenAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.PrintAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.SaveAction;

```

```

import com.mxgraph.swing.util.mxGraphActions;
import com.mxgraph.util.mxConstants;
import com.mxgraph.util.mxResources;

```

```

public class MenuBar extends JMenuBar
{

```

```

    /**
     *
     */
    private static final long serialVersionUID = 4060203894740766714L;

    public MenuBar(final AppMainWindow mainWindow)
    {
        JMenu menu = null;

        // Creates the file menu
        menu = add(new JMenu(mxResources.get("file")));
        populateFileMenu(menu, mainWindow);

        // Creates the edit menu
        menu = add(new JMenu(mxResources.get("edit")));
        populateEditMenu(menu, mainWindow);

        // Creates the conceptual menu
        menu = add(new JMenu(mxResources.get("conceptualModeling")));
        populateConceptualModelingMenu(menu, mainWindow);

        // Creates the logical menu
        menu = add(new JMenu(mxResources.get("logicalModeling")));
        populateLogicalModelingMenu(menu, mainWindow);

        // Creates the selection menu
        menu = add(new JMenu(mxResources.get("selection")));
        menu.setVisible(false); // it depends on an selected object to be visible
        populateSelectionMenu(menu, mainWindow);

        // Creates the window menu
        menu = add(new JMenu(mxResources.get("window")));
        populateWindowMenu(menu, mainWindow);

        // Creates the help menu
        menu = add(new JMenu(mxResources.get("help")));
    }

```



```

        populateHelpMenu(menu, mainWindow);
    }

    public void populateFileMenu(JMenu menu, final AppMainWindow mainWindow) {
        menu.add(mainWindow.bind(mxResources.get("newConceptual"), new
NewConceptualModelingAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/new_conceptual_small.png"));
        menu.add(mainWindow.bind(mxResources.get("newLogical"), new
NewLogicalModelingAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/new_logical_small.png"));
        menu.add(mainWindow.bind(mxResources.get("openFile"), new OpenAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/open_small.png"));
        menu.add(mainWindow.bind(mxResources.get("close"), new CloseAction()));

        menu.addSeparator();

        menu.add(mainWindow.bind(mxResources.get("save"), new SaveAction(false),
        "/ufsc/sisinf/brmodelo2all/ui/images/save_small.png"));
        menu.add(mainWindow.bind(mxResources.get("saveAs"), new SaveAction(true),
        "/ufsc/sisinf/brmodelo2all/ui/images/save_small.png"));

        menu.addSeparator();

        menu.add(mainWindow.bind(mxResources.get("print"), new PrintAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/print_small.png"));

        menu.addSeparator();

        menu.add(mainWindow.bind(mxResources.get("exit"), new ExitAction()));
    }

    public void populateEditMenu(JMenu menu, final AppMainWindow mainWindow) {
        menu.add(mainWindow.bind(mxResources.get("undo"), new HistoryAction(true),
        "/ufsc/sisinf/brmodelo2all/ui/images/undo_small.png"));
        menu.add(mainWindow.bind(mxResources.get("redo"), new HistoryAction(false),
        "/ufsc/sisinf/brmodelo2all/ui/images/redo_small.png"));

        menu.addSeparator();

        menu.add(mainWindow.bind(mxResources.get("cut"), TransferHandler
        .getCutAction(), "/ufsc/sisinf/brmodelo2all/ui/images/cut_small.png"));
        menu.add(mainWindow
        .bind(mxResources.get("copy"), TransferHandler.getCopyAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/copy_small.png"));
        menu.add(mainWindow.bind(mxResources.get("paste"), TransferHandler
        .getPasteAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/paste_small.png"));

        menu.addSeparator();

        menu.add(mainWindow.bind(mxResources.get("delete"), mxGraphActions
        .getDeleteAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/delete_small.png"));

        menu.addSeparator();

        menu.add(mainWindow.bind(mxResources.get("selectAll"), mxGraphActions
        .getSelectAllAction()));
        menu.add(mainWindow.bind(mxResources.get("selectNone"), mxGraphActions
        .getSelectNoneAction()));
    }

```

```
/**
```

```
* Adds menu items to the given format menu. This is factored out because
```

```

    * the format menu appears in the menubar and also in the popupmenu.
    */
    public static void populateFormatMenu(JMenu menu, AppMainWindow mainWindow)
    {
        menu.add(mainWindow.bind(mxResources.get("fillcolor"), new ColorAction(
            mxResources.get("fillcolor"), mxConstants.STYLE_FILLCOLOR),
            "/ufsc/sisinf/brmodelo2all/ui/images/fillcolor.gif"));
        menu.add(mainWindow.bind(mxResources.get("fontcolor"), new ColorAction(
            mxResources.get("fontcolor"), mxConstants.STYLE_FONTCOLOR),
            "/ufsc/sisinf/brmodelo2all/ui/images/fontcolor.gif"));
    }

    public void populateConceptualModelingMenu(JMenu menu, final AppMainWindow
mainWindow) {
        JMenu submenu = (JMenu) menu.add(new JMenu(mxResources.get("insertsObject")));
        submenu.add(mainWindow.bind(mxResources.get("entity"),
            new InsertConceptualObjectAction(mxResources.get("entity"),
            "/ufsc/sisinf/brmodelo2all/ui/images/rectangle.png"));
        submenu.add(mainWindow.bind(mxResources.get("relation"),
            new InsertConceptualObjectAction(mxResources.get("relation"),
            "/ufsc/sisinf/brmodelo2all/ui/images/rhombus.png"));
        submenu.add(mainWindow.bind(mxResources.get("selfRelation"),
            new InsertConceptualObjectAction(mxResources.get("selfRelation"),
            "/ufsc/sisinf/brmodelo2all/ui/images/self_related.png"));
        submenu.add(mainWindow.bind(mxResources.get("associativeEntity"),
            new
InsertConceptualObjectAction(mxResources.get("associativeEntity"),
            "/ufsc/sisinf/brmodelo2all/ui/images/entidade_associativa.png"));
        submenu.add(mainWindow.bind(mxResources.get("inheritance"),
            new InsertConceptualObjectAction(mxResources.get("inheritance"),
            "/ufsc/sisinf/brmodelo2all/ui/images/especializacao.png"));
        submenu.add(mainWindow.bind(mxResources.get("exclusiveInheritance"),
            new
InsertConceptualObjectAction(mxResources.get("exclusiveInheritance"),
            "/ufsc/sisinf/brmodelo2all/ui/images/especializacao_exclusiva.png"));
        submenu.add(mainWindow.bind(mxResources.get("nonExclusiveInheritance"),
            new
InsertConceptualObjectAction(mxResources.get("nonExclusiveInheritance"),
            "/ufsc/sisinf/brmodelo2all/ui/images/especializacao_naoexclusiva.png"));
        submenu.add(mainWindow.bind(mxResources.get("attribute"),
            new InsertConceptualObjectAction(mxResources.get("attribute"),
            "/ufsc/sisinf/brmodelo2all/ui/images/atributo.png"));
        submenu.add(mainWindow.bind(mxResources.get("identifierAttribute"),
            new
InsertConceptualObjectAction(mxResources.get("identifierAttribute"),
            "/ufsc/sisinf/brmodelo2all/ui/images/atributo_identificador.png"));
        submenu.add(mainWindow.bind(mxResources.get("composedAttribute"),
            new
InsertConceptualObjectAction(mxResources.get("composedAttribute"),
            "/ufsc/sisinf/brmodelo2all/ui/images/atributo_composto.png"));
        submenu.add(mainWindow.bind(mxResources.get("connector"),
            new InsertConceptualObjectAction(mxResources.get("connector"),
            "/ufsc/sisinf/brmodelo2all/ui/images/entity.png"));

        menu.addSeparator();

        menu.add(mainWindow.bind(mxResources.get("conceptualToLogical"),
            new ConvertConceptualToLogicalAction()));
    }

    public void populateLogicalModelingMenu(JMenu menu, final AppMainWindow mainWindow) {
        JMenu submenu = (JMenu) menu.add(new JMenu(mxResources.get("insertsObject")));

```

```

submenu.add(mainWindow.bind(mxResources.get("table"),
    new InsertLogicalObjectAction(mxResources.get("table"),
        "/ufsc/sisinf/brmodelo2all/ui/images/tabela.png"));
submenu.add(mainWindow.bind(mxResources.get("field"),
    new InsertLogicalObjectAction(mxResources.get("field"),
        "/ufsc/sisinf/brmodelo2all/ui/images/rectangle.png"));
submenu.add(mainWindow.bind(mxResources.get("primaryKey"),
    new InsertLogicalObjectAction(mxResources.get("primaryKey"),
        "/ufsc/sisinf/brmodelo2all/ui/images/chave_primaria.png"));
submenu.add(mainWindow.bind(mxResources.get("foreignKey"),
    new InsertLogicalObjectAction(mxResources.get("foreignKey"),
        "/ufsc/sisinf/brmodelo2all/ui/images/chave_estrangeira.png"));
submenu.add(mainWindow.bind(mxResources.get("separator"),
    new InsertLogicalObjectAction(mxResources.get("separator"),
        "/ufsc/sisinf/brmodelo2all/ui/images/separador.png"));
submenu.add(mainWindow.bind(mxResources.get("connector"),
    new InsertLogicalObjectAction(mxResources.get("connector"),
        "/ufsc/sisinf/brmodelo2all/ui/images/entity.png"));

menu.addSeparator();

menu.add(mainWindow.bind(mxResources.get("logicalToPhysical"),
    new ConvertLogicalToPhysicalAction()));
}

public void populateSelectionMenu(JMenu menu, final AppMainWindow mainWindow) {
    // Creates the format menu
    JMenu submenu = (JMenu) menu.add(new JMenu(mxResources.get("format")));
    populateFormatMenu(submenu, mainWindow);

    menu.addSeparator();

    menu.add(mainWindow.bind(mxResources.get("promoteToEntity"),
        new EntityPromotionAction(), null/*icon*/).setVisible(false);
    menu.add(mainWindow.bind(mxResources.get("promoteToAssociativeEntity"),
        new AssociativeEntityPromotionAction(),
        null/*icon*/).setVisible(false);
    menu.add(mainWindow.bind(mxResources.get("convertToExclusive"), null/*action*/,
        null/*icon*/).setVisible(false);
    menu.add(mainWindow.bind(mxResources.get("convertToOptional"), null/*action*/,
        null/*icon*/).setVisible(false);
}

public void populateWindowMenu(JMenu menu, final AppMainWindow mainWindow) {
    UIManager.LookAndFeelInfo[] lafs = UIManager.getInstalledLookAndFeels();

    for (int i = 0; i < lafs.length; i++)
    {
        final String clazz = lafs[i].getClassName();
        menu.add(new AbstractAction(lafs[i].getName())
        {
            /**
             *
             */
            private static final long serialVersionUID = -1959608552341208477L;

            public void actionPerformed(ActionEvent e)
            {
                mainWindow.setLookAndFeel(clazz);
            }
        });
    }
}

```

```

public void populateHelpMenu(JMenu menu, final AppMainWindow mainWindow) {
    JMenuItem item = menu.add(new JMenuItem(mxResources.get("about")));
    item.addActionListener(new ActionListener()
    {
        /*
         * (non-Javadoc)
         * @see
         java.awt.event.ActionListener#actionPerformed(java.awt.event.ActionEvent)
         */
        public void actionPerformed(ActionEvent e)
        {
            mainWindow.about();
        }
    });
}

/**
 * Adds menu items to the given shape menu. This is factored out because
 * the shape menu appears in the menubar and also in the popupmenu.
 */
//TODO APAGAR ESTE MÉTODO, NÃO DEVE MAIS EXISTIR
public static void populateShapeMenu(JMenu menu, AppMainWindow mainWindow)
{
    menu.add(mainWindow.bind(mxResources.get("home"), mxGraphActions
        .getHomeAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/house.gif"));

    menu.addSeparator();

    menu.add(mainWindow.bind(mxResources.get("exitGroup"), mxGraphActions
        .getExitGroupAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/up.gif"));
    menu.add(mainWindow.bind(mxResources.get("enterGroup"), mxGraphActions
        .getEnterGroupAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/down.gif"));

    menu.addSeparator();

    menu.add(mainWindow.bind(mxResources.get("group"), mxGraphActions
        .getGroupAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/group.gif"));
    menu.add(mainWindow.bind(mxResources.get("ungroup"), mxGraphActions
        .getUngroupAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/ungroup.gif"));

    menu.addSeparator();

    menu.add(mainWindow.bind(mxResources.get("removeFromGroup"), mxGraphActions
        .getRemoveFromParentAction()));

    menu.add(mainWindow.bind(mxResources.get("updateGroupBounds"),
        mxGraphActions.getUpdateGroupBoundsAction()));

    menu.addSeparator();

    menu.add(mainWindow.bind(mxResources.get("collapse"), mxGraphActions
        .getCollapseAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/collapse.gif"));
    menu.add(mainWindow.bind(mxResources.get("expand"), mxGraphActions
        .getExpandAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/expand.gif"));

    menu.addSeparator();

    menu.add(mainWindow.bind(mxResources.get("toBack"), mxGraphActions

```

```

        .getToBackAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/toback.gif"));
menu.add(mainWindow.bind(mxResources.get("toFront"), mxGraphActions
        .getToFrontAction(),
        "/ufsc/sisinf/brmodelo2all/ui/images/tofront.gif"));

menu.addSeparator();

JMenu submenu = (JMenu) menu.add(new JMenu(mxResources.get("align")));

submenu.add(mainWindow.bind(mxResources.get("left"), new AlignCellsAction(
        mxConstants.ALIGN_LEFT,
        "/ufsc/sisinf/brmodelo2all/ui/images/alignleft.gif"));
submenu.add(mainWindow.bind(mxResources.get("center"),
        new AlignCellsAction(mxConstants.ALIGN_CENTER,
        "/ufsc/sisinf/brmodelo2all/ui/images/aligncenter.gif"));
submenu.add(mainWindow.bind(mxResources.get("right"), new AlignCellsAction(
        mxConstants.ALIGN_RIGHT),
        "/ufsc/sisinf/brmodelo2all/ui/images/alignright.gif"));

submenu.addSeparator();

submenu.add(mainWindow.bind(mxResources.get("top"), new AlignCellsAction(
        mxConstants.ALIGN_TOP,
        "/ufsc/sisinf/brmodelo2all/ui/images/aligntop.gif"));
submenu.add(mainWindow.bind(mxResources.get("middle"),
        new AlignCellsAction(mxConstants.ALIGN_MIDDLE,
        "/ufsc/sisinf/brmodelo2all/ui/images/alignmiddle.gif"));
submenu.add(mainWindow.bind(mxResources.get("bottom"),
        new AlignCellsAction(mxConstants.ALIGN_BOTTOM,
        "/ufsc/sisinf/brmodelo2all/ui/images/alignbottom.gif"));

menu.addSeparator();

menu
        .add(mainWindow.bind(mxResources.get("autosize"),
                new AutosizeAction()));
    }
}

package ufsc.sisinf.brmodelo2all.util;

import java.util.MissingResourceException;
import java.util.ResourceBundle;

public class Messages {
    private static final String BUNDLE_NAME = "ufsc.sisinf.brmodelo2all.ui.resources.messages";
    //$NON-NLS-1$

    private static final ResourceBundle RESOURCE_BUNDLE = ResourceBundle
        .getBundle(BUNDLE_NAME);

    private Messages() {
    }

    public static String getString(String key) {
        try {
            return RESOURCE_BUNDLE.getString(key);
        } catch (MissingResourceException e) {
            return '!' + key + '!';
        }
    }
}
}

```

```

package ufsc.sisinf.brmodelo2all.model;

import java.text.NumberFormat;
import java.util.ArrayList;
import java.util.List;

import ufsc.sisinf.brmodelo2all.model.objects.AssociativeRelationObject;
import ufsc.sisinf.brmodelo2all.model.objects.EntityObject;
import ufsc.sisinf.brmodelo2all.model.objects.ModelingObject;

import com.mxgraph.model.mxCell;
import com.mxgraph.model.mxGeometry;
import com.mxgraph.model.mxICell;
import com.mxgraph.util.mxResources;
import com.mxgraph.view.mxGraph;

public class Modeling extends mxGraph {
    public static final NumberFormat numberFormat = NumberFormat.getInstance();

    /**
     * Holds the edge to be used as a template for inserting new edges.
     */
    protected Object edgeTemplate;

    /**
     * Custom graph that defines the alternate edge style to be used when the
     * middle control point of edges is double clicked (flipped).
     */
    public Modeling() {
        setAlternateEdgeStyle("edgeStyle=mxEdgeStyle.ElbowConnector;elbow=vertical");
        setCellsDisconnectable(false);
        setDropEnabled(false);
    }

    /**
     * Sets the edge template to be used to inserting edges.
     */
    public void setEdgeTemplate(Object template) {
        edgeTemplate = template;
    }

    /**
     * Prints out some useful information about the cell in the tooltip.
     */
    public String getToolTipForCell(Object cell) {
        String tip = null;

        if (cell instanceof mxICell) {
            Object object = ((mxICell) cell).getValue();
            if (object instanceof AssociativeRelationObject) {
                object = ((mxCell)cell).getParent().getValue();
                cell = ((mxCell)cell).getParent();
            }

            if (object instanceof ModelingObject) {
                mxGeometry geo = getModel().getGeometry(cell);
                tip = "<html>";
                ModelingObject objectModeling = (ModelingObject) object;
                tip += "<b>" + mxResources.get("objectProperties")
                    + "</b><br>";
                tip += objectModeling.getToolTip();

                if (getModel().isVertex(cell)) {
                    if (geo != null) {

```

```

        tip += mxResources.get("position") + "(x,y): (";
        tip += numberFormat.format(geo.getX()) + ",";
        tip += numberFormat.format(geo.getY()) + "<br>";
        tip += mxResources.get("width") + ": "
        tip += numberFormat.format(geo.getWidth()) + "<br>";
        tip += mxResources.get("height") + ": "
        tip += numberFormat.format(geo.getHeight());
    }
}
tip += "</html>";
}
}
return tip;
}

// TODO algoritmo para gerar tooltip de conectores original. guardado para consultar se necessário.
apagar no final do projeto
public void oldToolTipForEdge() {
//      mxCellState state = getView().getState(cell);
//      tip += "points={";
//
//      if (geo != null) {
//          List<mxPoint> points = geo.getPoints();
//
//          if (points != null) {
//              Iterator<mxPoint> it = points.iterator();
//
//              while (it.hasNext()) {
//                  mxPoint point = it.next();
//                  tip += "[x=" + numberFormat.format(point.getX())
//                      + ",y=" + numberFormat.format(point.getY())
//                      + "],";
//              }
//
//              tip = tip.substring(0, tip.length() - 1);
//          }
//      }
//
//      tip += "<br>";
//      tip += "absPoints={";
//
//      if (state != null) {
//
//          for (int i = 0; i < state.getAbsolutePointCount(); i++) {
//              mxPoint point = state.getAbsolutePoint(i);
//              tip += "[x=" + numberFormat.format(point.getX()) + ",y="
//                  + numberFormat.format(point.getY()) + "],";
//          }
//
//          tip = tip.substring(0, tip.length() - 1);
//      }
//
//      tip += "}";
}

/**
 * Overrides the method to use the currently selected edge template for new
 * edges.
 *
 * @param graph
 * @param parent
 * @param id

```

```

* @param value
* @param source
* @param target
* @param style
* @return
*/
public Object createEdge(Object parent, String id, Object value,
                        Object source, Object target, String style) {

    return super.createEdge(parent, id, value, source, target, style);
}

@Override
public void setSelectionCell(Object cell) {
    if (((mxCell) cell).getStyle() == "straight")
        super.setSelectionCell(null);
    else if (((mxCell) cell).getStyle() != "associativeRelation")
        super.setSelectionCell(cell);
    else
        super.setSelectionCell(((mxCell)cell).getParent());
}

@Override
public boolean isCellSelectable(Object cell) {
    return ((mxCell) cell).getStyle() != "straight";
}

@Override
public Object[] removeCells(Object[] cells, boolean includeEdges) {

    if (cells == null)
        cells = getDeletableCells(getSelectionCells());

    removeReferencesFromParents(cells);

    List<Object> allRelatedCells = new ArrayList<Object>();
    addAllRelatedCells(cells, allRelatedCells);

    return super.removeCells(allRelatedCells.toArray(), includeEdges);
}

public void removeReferencesFromParents(Object[] cells) {
    for (int i = 0; i < cells.length; i++) {
        ModelingObject modelingObject =
(ModelingObject)((mxCell)cells[i]).getValue();
        if (modelingObject != null)
            modelingObject.removeReferencesFromParents();
    }
}

public void addAllRelatedCells(Object[] cells, List<Object> allRelatedCells) {
    for (int i = 0; i < cells.length; i++) {
        mxCell cell = (mxCell) cells[i];
        if (cell.getValue() instanceof ModelingObject) {
            ModelingObject modelingObject = (ModelingObject)cell.getValue();

            if (!(cell.getValue() instanceof EntityObject) ||
                ((cell.getValue() instanceof EntityObject) &&
                ((EntityObject)cell.getValue()).getParentObject() == null)){

                addAllRelatedCells(modelingObject.getChildObjects().toArray(), allRelatedCells);
                allRelatedCells.add(cells[i]);
            }
        }
    }
}

```



```
    }  
  }  
}
```

```
package ufsc.sisinf.brmodelo2all.model;
```

```
import java.awt.Color;  
import java.awt.Point;  
import java.awt.Rectangle;  
import java.awt.event.MouseEvent;  
import java.util.EventObject;
```

```
import org.w3c.dom.Document;
```

```
import ufsc.sisinf.brmodelo2all.app.BrModelo2All;  
import ufsc.sisinf.brmodelo2all.control.ModelingHandler;  
import ufsc.sisinf.brmodelo2all.model.objects.ModelingObject;  
import ufsc.sisinf.brmodelo2all.ui.AppMainWindow;  
import com.mxgraph.io.mxCodec;  
import com.mxgraph.model.mxCell;  
import com.mxgraph.model.mxICell;  
import com.mxgraph.model.mxIGraphModel;  
import com.mxgraph.swing.mxGraphComponent;  
import com.mxgraph.swing.handler.mxGraphHandler;  
import com.mxgraph.swing.view.mxInteractiveCanvas;  
import com.mxgraph.util.mxUtils;  
import com.mxgraph.view.mxGraph;
```

```
public class ModelingComponent extends mxGraphComponent {
```

```
    private static final long serialVersionUID = -6833603133512882012L;
```

```
    protected final AppMainWindow mainWindow;
```

```
    /**  
     *  
     * @param graph  
     */
```

```
    public ModelingComponent(mxGraph graph, final AppMainWindow mainWindow) {  
        super(graph);
```

```
        this.mainWindow = mainWindow;
```

```
        // Sets switches typically used in an editor  
        setPageVisible(false);  
        setGridVisible(false);  
        setToolTips(true);  
        getConnectionHandler().setCreateTarget(true);  
        // impedir que faça uma marca ao colocar o mouse sobre um vertex  
        getConnectionHandler().getMarker().setEnabled(false);
```

```
        // Loads the default stylesheet from an external file  
        mxCodec codec = new mxCodec();  
        Document doc = mxUtils.loadDocument(BrModelo2All.class.getResource(  
            "/ufsc/sisinf/brmodelo2all/ui/resources/default-style.xml")  
            .toString());  
        codec.decode(doc.getDocumentElement(), graph.getStylesheet());
```

```
        // Sets the background to white  
        getViewport().setOpaque(false);  
        setBackground(Color.WHITE);
```

```
    }
```

```
    public AppMainWindow getMainWindow() {  
        return mainWindow;
```

```

}

@Override
protected mxGraphHandler createGraphHandler() {
    return new ModelingHandler(this);
}

/**
 * Overrides drop behaviour to set the cell style if the target is not a
 * valid drop target and the cells are of the same type (eg. both vertices
 * or both edges).
 */
public Object[] importCells(Object[] cells, double dx, double dy,
    Object target, Point location) {
    if (target == null && cells.length == 1 && location != null) {
        target = getCellAt(location.x, location.y);

        if (target instanceof mxICell && cells[0] instanceof mxICell) {
            mxICell targetCell = (mxICell) target;
            mxICell dropCell = (mxICell) cells[0];

            if (targetCell.isVertex() == dropCell.isVertex()
                || targetCell.isEdge() == dropCell.isEdge()) {
                mxIGraphModel model = graph.getModel();
                model.setStyle(target, model.getStyle(cells[0]));
                graph.setSelectionCell(null/* target */);

                return null;
            }
        }
    }

    return super.importCells(cells, dx, dy, target, location);
}

/**
 *
 */
public mxInteractiveCanvas createCanvas() {
    return new ufsc.sisinf.br.modelo2all.modelo.shapes.ShapesCanvas();
}

/**
 *
 */
public void startEditingAtCell(Object cell, EventObject evt) {
    if (cell == null) {
        cell = graph.getSelectionCell();

        if (cell != null && !graph.isCellEditable(cell)) {
            cell = null;
        }
    }

    if (cell != null) {
        mainWindow.properties(cell, this);
    }
}

@Override
public Object[] selectRegion(Rectangle rect, MouseEvent e) {
    Object[] cells = getCells(rect);

    if (cells.length == 0)
        mainWindow.updateSelectionMenu(null);
}

```

```

        else if (cells.length == 1)

mainWindow.updateSelectionMenu((ModelingObject)((mxCell)cells[0]).getValue());
        else
            mainWindow.updateSelectionMenu(true);

        return super.selectRegion(rect, e);
    }

/**
 *
 */
protected void installDoubleClickHandler()
{
}

}

package ufsc.sisinf.brmodelo2all.model.objects;

import java.io.Serializable;
import java.util.Iterator;
import java.util.List;
import java.util.ArrayList;

import javax.swing.JComponent;

import com.mxgraph.model.mxCell;
import com.mxgraph.util.mxResources;

import ufsc.sisinf.brmodelo2all.util.AppConstants;

public class ModelingObject implements Serializable{

    /**
     *
     */
    private static final long serialVersionUID = -7570012929655893724L;

    private List<Object> childObjects = new ArrayList<Object>();

    private Object parentObject = null;

    private List<RelationObject> relations = new ArrayList<RelationObject>();

    private String name;
    // observações
    private String notes;

    private final int NUMBER_OF_ATTRIBUTES = 2;

    public ModelingObject(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {

```

```

        return name;
    }

    public void setNotes(String notes) {
        this.notes = notes;
    }

    public String getNotes() {
        return notes;
    }

    public void setChildObjects(List<Object> childObjects) {
        this.childObjects = childObjects;
    }

    public List<Object> getChildObjects() {
        return childObjects;
    }

    public void addChildObject(Object modelingObject) {
        childObjects.add(modelingObject);
    }

    public void removeChildObject(Object modelingObject) {
        Object objectToRemove = null;
        if (modelingObject instanceof ModelingObject) {
            Iterator<Object> iterator = childObjects.iterator();
            while (iterator.hasNext() && objectToRemove == null) {
                Object object = iterator.next();

                if (((ModelingObject)((mxCell)object).getValue() ==
(ModelingObject)modelingObject)
                    objectToRemove = object;
            }

        } else
            objectToRemove = modelingObject;

        childObjects.remove(objectToRemove);
    }

    public void addRelation(RelationObject relation) {
        relations.add(relation);
    }

    public void removeRelation(RelationObject relation) {
        relations.remove(relation);
    }

    public Iterator<RelationObject> getRelationsIterator() {
        return relations.iterator();
    }

    public void setParentObject(Object parentObject) {
        this.parentObject = parentObject;
    }

    public Object getParentObject() {
        return parentObject;
    }

    public int attributesCount(){
        return NUMBER_OF_ATTRIBUTES;
    }

```

```

public int getIndexForComponents() {
    return 0;
}

public void getAttributes(int types[], String names[], String values[], boolean fieldsEnabled[]) {
    int i = getIndexForComponents();

    types[i] = AppConstants.TEXT_FIELD;
    names[i] = getNameLabel();
    values[i] = name;
    fieldsEnabled[i++] = true;
    types[i] = AppConstants.TEXT_FIELD;
    names[i] = mxResources.get("notes");
    values[i] = notes;
    fieldsEnabled[i] = true;
}

public String getNameLabel() {
    return mxResources.get("name");
}

public void setAttributes(String values[]) {
    int i = getIndexForComponents();

    setName(values[i++]);
    setNotes(values[i]);
}

public int windowHeight() {
    return 180;
}

public String[] getComboValues(String name){
    return null;
}

public String getToolTip() {
    return mxResources.get("name") + ": " + name + "<br>";
}

public String getStyle() {
    return null;
}

public void createHandlers(JComponent[] components) {
}

public void removeReferencesFromParents() {
}
}

package ufsc.sisinf.brmodelo2all.ui;

import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.datatransfer.DataFlavor;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.util.ArrayList;
import java.util.List;

import javax.swing.ImageIcon;

```

```

import javax.swing.JComponent;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.TransferHandler;

import com.mxgraph.model.mxCell;
import com.mxgraph.model.mxGeometry;
import com.mxgraph.swing.util.mxGraphTransferable;
import com.mxgraph.util.mxEvent;
import com.mxgraph.util.mxEventObject;
import com.mxgraph.util.mxEventSource;
import com.mxgraph.util.mxPoint;
import com.mxgraph.util.mxRectangle;
import com.mxgraph.util.mxResources;
import com.mxgraph.util.mxEventSource.mxEventListener;

public class ModelingPalette extends JPanel
{
    private static final long serialVersionUID = 7771113885935187066L;

    protected mxEventSource eventSource = new mxEventSource(this);
    protected List<mxGraphTransferable> transferables = new ArrayList<mxGraphTransferable>();
    protected JLabel selectedEntry = null;
    protected final AppMainWindow mainWindow;

    @SuppressWarnings("serial")
    public ModelingPalette(final AppMainWindow mainWindow)
    {
        this.mainWindow = mainWindow;

        setLayout(new FlowLayout(FlowLayout.LEADING, 0, 0));

        // Clears the current selection when the background is clicked
        addMouseListener(new MouseListener()
        {

            public void mousePressed(MouseEvent e)
            {
                clearSelection();
            }

            public void mouseClicked(MouseEvent e)
            {
            }

            public void mouseEntered(MouseEvent e)
            {
            }

            public void mouseExited(MouseEvent e)
            {
            }

            public void mouseReleased(MouseEvent e)
            {
            }

        });

        // Shows a nice icon for drag and drop but doesn't import anything
        setTransferHandler(new TransferHandler()
        {

            public boolean canImport(JComponent comp, DataFlavor[] flavors)
            {

                return true;
            }

        });
    }
}

```

```

        }
    });
}

public JLabel getSelectedEntry() {
    return selectedEntry;
}

public void clearSelection()
{
    setSelectionEntry(null, null);
}

public void setSelectionEntry(JLabel entry, mxGraphTransferable t)
{
    JLabel previous = selectedEntry;
    if (mainWindow.getCurrentEditor() == null ||
        previous == entry) {
        selectedEntry = null;
    } else {
        selectedEntry = entry;
    }

    if (previous != null)
    {
        previous.setBorder(null);
        previous.setOpaque(false);
    }

    if (selectedEntry != null)
    {
        selectedEntry.setBorder(ShadowBorder.getSharedInstance());
        selectedEntry.setOpaque(true);
    }

    eventSource.fireEvent(new mxEventObject(mxEvent.SELECT, "entry",
        selectedEntry, "transferable", t, "previous", previous));
}

public void setPreferredWidth(int width)
{
    int cols = Math.max(1, width / 55);
    setPreferredSize(new Dimension(width, (getComponentCount() * 55 / cols)/* + 30*/));
    revalidate();
}

/**
 *
 * @param name
 * @param icon
 * @param style
 * @param width
 * @param height
 * @param value
 */
public void addEdgeTemplate(final String name, ImageIcon icon,
    String style, int width, int height, Object value)
{
    mxGeometry geometry = new mxGeometry(0, 0, width, height);
    geometry.setTerminalPoint(new mxPoint(0, height), true);
    geometry.setTerminalPoint(new mxPoint(width, 0), false);
    geometry.setRelative(true);
}

```

```

        mxCell cell = new mxCell(value, geometry, style);
        cell.setEdge(true);

        addTemplate(name, icon, cell);
    }

/**
 *
 * @param name
 * @param icon
 * @param style
 * @param width
 * @param height
 * @param value
 */
public void addTemplate(final String name, ImageIcon icon, String style,
                       int width, int height, Object value)
{
    mxCell cell = new mxCell(value, new mxGeometry(0, 0, width, height),
                              style);
    cell.setVertex(true);
    addTemplate(name, icon, cell);
}

/**
 *
 * @param name
 * @param icon
 * @param cell
 */
public void addTemplate(final String name, ImageIcon icon, mxCell cell)
{
    mxRectangle bounds = (mxGeometry) cell.getGeometry().clone();
    final mxGraphTransferable t = new mxGraphTransferable(
        new Object[] { cell }, bounds);
    transferables.add(t);

    // Scales the image if it's too large for the library
    if (icon != null)
    {
        if (icon.getIconWidth() > 32 || icon.getIconHeight() > 32)
        {
            icon = new ImageIcon(icon.getImage().getScaledInstance(32, 32,
                0));
        }
    }

    final JLabel entry = new JLabel(icon);
    entry.setPreferredSize(new Dimension(40,40));
    entry.setBackground(ModelingPalette.this.getBackground().brighter());
    entry.setFont(new Font(entry.getFont().getFamily(), 0, 10));

    entry.setVerticalTextPosition(JLabel.BOTTOM);
    entry.setHorizontalTextPosition(JLabel.CENTER);
    entry.setIconTextGap(0);

    entry.setToolTipText(name);

    entry.addMouseListener(new MouseListener()
    {
        public void mousePressed(MouseEvent e)
        {
            setSelectionEntry(entry, t);
        }
    }

```



```

        public void mouseClicked(MouseEvent e)
        {
        }

        public void mouseEntered(MouseEvent e)
        {
            setStatusBarMessage(entry.getToolTipText());
        }

        public void mouseExited(MouseEvent e)
        {
            setStatusBarMessage(null);
        }

        public void mouseReleased(MouseEvent e)
        {
        }

    });

    add(entry);
}

public mxGraphTransferable getTransferable(int index){
    return transferables.get(index);
}

/**
 * @param eventName
 * @param listener
 * @see com.mxgraph.util.mxEventSource#addListener(java.lang.String,
com.mxgraph.util.mxEventSource.mxEventListener)
 */
public void addListener(String eventName, mxEventListener listener)
{
    eventSource.addListener(eventName, listener);
}

/**
 * @return whether or not event are enabled for this palette
 * @see com.mxgraph.util.mxEventSource#isEventsEnabled()
 */
public boolean isEventsEnabled()
{
    return eventSource.isEventsEnabled();
}

/**
 * @param listener
 * @see
com.mxgraph.util.mxEventSource#removeListener(com.mxgraph.util.mxEventSource.mxEventListener)
 */
public void removeListener(mxEventListener listener)
{
    eventSource.removeListener(listener);
}

/**
 * @param eventName
 * @param listener
 * @see com.mxgraph.util.mxEventSource#removeListener(java.lang.String,
com.mxgraph.util.mxEventSource.mxEventListener)
 */
public void removeListener(mxEventListener listener, String eventName)

```

```

    {
        eventSource.removeListener(listener, eventName);
    }

/**
 * @param eventsEnabled
 * @see com.mxgraph.util.mxEventSource#setEventsEnabled(boolean)
 */
public void setEventsEnabled(boolean eventsEnabled)
{
    eventSource.setEventsEnabled(eventsEnabled);
}

public void setStatusBarItemMessage(String message) {

    if (message == null) {
        mainWindow.status(null, null);
        return;
    }

    String status = "";

    if (mainWindow.getSelectedEntry() == null) {
        if (message == mxResources.get("entity"))
            status = mxResources.get("insertsEntity");
        else if (message == mxResources.get("relation"))
            status = mxResources.get("insertsRelation");
        else if (message == mxResources.get("selfRelation"))
            status = mxResources.get("insertsSelfRelation");
        else if (message == mxResources.get("inheritance"))
            status = mxResources.get("insertsInheritance");
        else if (message == mxResources.get("exclusiveInheritance"))
            status = mxResources.get("insertsExclusiveInheritance");
        else if (message == mxResources.get("nonExclusiveInheritance"))
            status = mxResources.get("insertsNonExclusiveInheritance");
        else if (message == mxResources.get("associativeEntity"))
            status = mxResources.get("insertsAssociativeEntity");
        else if (message == mxResources.get("attribute"))
            status = mxResources.get("insertsAttribute");
        else if (message == mxResources.get("identifierAttribute"))
            status = mxResources.get("insertsIdentifierAttribute");
        else if (message == mxResources.get("composedAttribute"))
            status = mxResources.get("insertsComposedAttribute");
        else if (message == mxResources.get("connector"))
            status = mxResources.get("insertsConnector");
        else if (message == mxResources.get("table"))
            status = mxResources.get("insertsTable");
        else if (message == mxResources.get("field"))
            status = mxResources.get("insertsField");
        else if (message == mxResources.get("primaryKey"))
            status = mxResources.get("insertsPrimaryKey");
        else if (message == mxResources.get("foreignKey"))
            status = mxResources.get("insertsForeignKey");
    }
    mainWindow.status(null, status);
    mainWindow.statusBar.setFont(new Font(mainWindow.statusBar.getFont().getFamily(),
Font.BOLD, 11));
}

}

package ufsc.sisinf.br.modelo2all.ui;

import javax.swing.JMenu;

```

```

import javax.swing.JPopupMenu;
import javax.swing.TransferHandler;

import ufsc.sisinf.brmodelo2all.ui.CommandActions.HistoryAction;
import com.mxgraph.swing.util.mxGraphActions;
import com.mxgraph.util.mxResources;

public class PopupMenu extends JPopupMenu {

    /**
     *
     */
    private static final long serialVersionUID = -3132749140550242191L;

    public PopupMenu(AppMainWindow mainWindow)
    {
        boolean selected = !mainWindow.getCurrentEditor().getGraphComponent().getGraph()
            .isSelectionEmpty();

        add(mainWindow.bind(mxResources.get("undo"), new HistoryAction(true),
            "/ufsc/sisinf/brmodelo2all/ui/images/undo_small.png"));

        addSeparator();

        add(
            mainWindow.bind(mxResources.get("cut"), TransferHandler
                .getCutAction(),
                "/ufsc/sisinf/brmodelo2all/ui/images/cut_small.png"))
            .setEnabled(selected);

        add(
            mainWindow.bind(mxResources.get("copy"), TransferHandler
                .getCopyAction(),
                "/ufsc/sisinf/brmodelo2all/ui/images/copy_small.png"))
            .setEnabled(selected);

        add(mainWindow.bind(mxResources.get("paste"), TransferHandler
            .getPasteAction(),
            "/ufsc/sisinf/brmodelo2all/ui/images/paste_small.png"));

        addSeparator();

        add(
            mainWindow.bind(mxResources.get("delete"), mxGraphActions
                .getDeleteAction(),
                "/ufsc/sisinf/brmodelo2all/ui/images/delete_small.png"))
            .setEnabled(selected);

        addSeparator();

        // Creates the format menu
        JMenu menu = (JMenu) add(new JMenu(mxResources.get("format")));
        MenuBar.populateFormatMenu(menu, mainWindow);
        menu.setEnabled(selected);

        addSeparator();

        add(mainWindow.bind(mxResources.get("promoteToAssociativeEntity"), null/*action*/,
            null/*icon*/));

        addSeparator();

        add(mainWindow.bind(mxResources.get("objectProperties"),
            null/*action*/)).setEnabled(selected);
    }
}

```

```

package ufsc.sisinf.brmodelo2all.ui;

import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;

import javax.swing.BorderFactory;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JComponent;
import javax.swing.JDialog;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JRootPane;
import javax.swing.JTextField;
import javax.swing.KeyStroke;

import ufsc.sisinf.brmodelo2all.model.ModelingComponent;
import ufsc.sisinf.brmodelo2all.model.objects.AssociativeRelationObject;
import ufsc.sisinf.brmodelo2all.model.objects.ModelingObject;
import ufsc.sisinf.brmodelo2all.util.AppConstants;

import com.mxgraph.model.mxCell;
import com.mxgraph.util.mxResources;

public class PropertiesWindow extends JDialog {

    private ModelingObject modelingObject;
    private JComponent[] fields;
    private mxCell cellObject;
    protected final ModelingComponent modelingComponent;

    /**
     *
     */
    private static final long serialVersionUID = -8803589863674906236L;
    public static final int WINDOW_WIDTH = 180;

    public PropertiesWindow(Frame owner, Object cell, final ModelingComponent
modelingComponent) {
        super(owner);

        this.modelingComponent = modelingComponent;
        String title = mxResources.get("objectProperties");
        setTitle(title);
        setLayout(new BorderLayout());
        cellObject = (mxCell) cell;
        modelingObject = (ModelingObject) cellObject.getValue();
        if(modelingObject instanceof AssociativeRelationObject) {
            cellObject = (mxCell)((mxCell)cell).getParent();
            modelingObject = (ModelingObject) cellObject.getValue();
        }

        fields = new JComponent[modelingObject.attributesCount()];

        // montar as propriedades do objeto
        JPanel attributesPanel = createAttributesPanel();
        getContentPane().add(attributesPanel, BorderLayout.NORTH);
        if (fields[0] instanceof JTextField)

```

```

        ((JTextField)fields[0]).selectAll();

// panel dos botões
JPanel buttonsPanel = createButtonsPanel();
getContentPane().add(buttonsPanel, BorderLayout.CENTER);

setResizable(false);
setSize(WINDOW_WIDTH, getWindowHeight());
}

public int getWindowHeight(){
    return modelingObject.windowHeight();
}

protected JPanel createAttributesPanel() {
    JPanel p = new JPanel();
    int count = modelingObject.attributesCount();
    p.setLayout(new BorderLayout(p, BorderLayout.Y_AXIS));
    p.setBorder(BorderFactory.createEmptyBorder(0, 6, 0, 6));

    int fieldsTypes[] = new int[count];
    String fieldsNames[] = new String[count];
    String fieldsValues[] = new String[count];
    boolean fieldsEnabled[] = new boolean[count];
    modelingObject.getAttributes(fieldsTypes, fieldsNames, fieldsValues, fieldsEnabled);

    for (int i = 0; i < count; i++) {
        fillAttributeComponents(p, fieldsNames[i], fieldsValues[i], fieldsTypes[i],
fieldsEnabled[i], i);
    }

    modelingObject.createHandlers(fields);

    return p;
}

protected void fillAttributeComponents(JPanel panel, String name, String value, int type, boolean
enabled, int fieldIndex) {
    boolean usesCaption = true;
    JComponent field;
    switch(type) {
        case AppConstants.TEXT_FIELD:
            JTextField text = new JTextField(value, 10);
            text.setSize(150, 5);
            text.setEnabled(enabled);
            field = text;
            break;

        case AppConstants.CHECK_BOX:
            boolean selected = value == "true" ? true : false;
            usesCaption = false;
            JCheckBox checkBox = new JCheckBox(name, selected);
            checkBox.setEnabled(enabled);
            field = checkBox;
            break;

        case AppConstants.COMBO_BOX:
            String values[] = modelingObject.getComboValues(name);
            JComboBox combo = new JComboBox(values);
            combo.setSelectedItem(value);
            combo.setEnabled(enabled);
            field = combo;
            break;

        default:

```

```

        usesCaption = false;
        field = new JLabel(value);
    }

    if (usesCaption){
        JLabel caption = new JLabel(name);
        caption.setBorder(BorderFactory.createEmptyBorder(4, 0, 0, 0));
        if (type == AppConstants.COMBO_BOX) {
            JPanel innerPanel = new JPanel();
            innerPanel.setLayout(new BorderLayout());
            innerPanel.setBorder(BorderFactory.createEmptyBorder(4, 0, 0, 0));
            innerPanel.add(caption, BorderLayout.WEST);
            innerPanel.add(field, BorderLayout.EAST);
            panel.add(innerPanel);
        } else {
            JPanel innerPanel = new JPanel();
            innerPanel.setLayout(new BorderLayout());
            innerPanel.setBorder(BorderFactory.createEmptyBorder(4, 0, 0, 0));
            innerPanel.add(caption, BorderLayout.WEST);
            panel.add(innerPanel);
            innerPanel = new JPanel();
            innerPanel.setLayout(new BorderLayout());
            innerPanel.setBorder(BorderFactory.createEmptyBorder(0, 0, 0, 0));
            innerPanel.add(field, BorderLayout.CENTER);
            panel.add(innerPanel);
        }
    } else {
        JPanel innerPanel = new JPanel();
        innerPanel.setLayout(new BorderLayout());
        innerPanel.setBorder(BorderFactory.createEmptyBorder(4, 0, 0, 0));
        innerPanel.add(field, BorderLayout.WEST);
        panel.add(innerPanel);
    }

    fields[fieldIndex] = field;
}

protected JPanel createButtonsPanel() {
    JPanel p = new JPanel(new FlowLayout(FlowLayout.CENTER));
    p.setBorder(BorderFactory.createEmptyBorder(8, 0, 0, 0));
    // Adds OK button to close window
    JButton okButton = new JButton("Ok");
    okButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            saveModifications();
            setVisible(false);
        }
    });

    // Adds OK button to close window
    JButton cancelButton = new JButton("Cancelar");
    cancelButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            setVisible(false);
        }
    });

    p.add(okButton);
    p.add(cancelButton);

    // Sets default button for enter key
    getRootPane().setDefaultButton(okButton);
}

```

```

        }
        return p;
    }

    public void saveModifications() {
        int count = modelingObject.attributesCount();

        int fieldsTypes[] = new int[count];
        String fieldsNames[] = new String[count];
        String fieldsValues[] = new String[count];
        boolean fieldsEnabled[] = new boolean[count];
        modelingObject.getAttributes(fieldsTypes, fieldsNames, fieldsValues, fieldsEnabled);

        String newValues[] = new String[count];
        for (int i = 0; i < count; i++) {

            switch(fieldsTypes[i]) {
                case AppConstants.TEXT_FIELD:
                    newValues[i] = ((JTextField)fields[i]).getText();
                    break;

                case AppConstants.CHECK_BOX:
                    newValues[i] = ((JCheckBox)fields[i]).isSelected() ? "true" :
"false";
                    break;

                case AppConstants.COMBO_BOX:
                    newValues[i] =
(String)((JComboBox)fields[i]).getSelectedObjects()[0];
                    break;
            }
        }

        boolean modified = false;
        for (int i = 0; i < count && !modified; i++) {
            modified = newValues[i] != fieldsValues[i];
        }

        if (modified) {
            modelingObject.setAttributes(newValues);

            modelingComponent.getGraph().getModel().beginUpdate();
            try {
                Object[] objects = { cellObject };
                modelingComponent.getGraph().setCellStyle(modelingObject.getStyle(),
objects);

                modelingComponent.getGraph().repaint(cellObject.getGeometry().getAlternateBounds());

            } finally {
                modelingComponent.getGraph().getModel().endUpdate();
            }
        }
    }
    /**
     * Overrides {@link JDialog#createRootPane()} to return a root pane that
     * hides the window when the user presses the ESCAPE key.O
     */
    protected JRootPane createRootPane() {
        KeyStroke stroke = KeyStroke.getKeyStroke(KeyEvent.VK_ESCAPE, 0);
        JRootPane rootPane = new JRootPane();
        rootPane.registerKeyboardAction(new ActionListener() {
            public void actionPerformed(ActionEvent actionEvent) {
                setVisible(false);
            }
        }, stroke, JComponent.WHEN_IN_FOCUSED_WINDOW);
    }
}

```

```

        }
        return rootPane;
    }
}

package ufsc.sisinf.br.modelo2all.model.objects;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import com.mxgraph.model.mxCell;

public class RelationObject extends ModelingObject {

    private static final long serialVersionUID = -4379194659559624895L;

    private List<ModelingObject> relatedObjects = new ArrayList<ModelingObject>();

    private final int NUMBER_OF_ATTRIBUTES = 0;

    public RelationObject(String name) {
        super(name);
    }

    public RelationObject(String name, Object parentObject) {
        super(name);
        setParentObject(parentObject);
    }

    public void addRelatedObject(ModelingObject object) {
        relatedObjects.add(object);
    }

    public void removeRelatedObject(ModelingObject object) {
        relatedObjects.remove(object);
    }

    public Iterator<ModelingObject> getRelatedObjectsIterator() {
        return relatedObjects.iterator();
    }

    public List<ModelingObject> getRelatedObjects() {
        return relatedObjects;
    }

    public int attributesCount() {
        return super.attributesCount() + NUMBER_OF_ATTRIBUTES;
    }

    public void getAttributes(int types[], String names[], String values[], boolean enabled[]) {
        super.getAttributes(types, names, values, enabled);
    }

    public int windowHeight() {
        return super.windowHeight();
    }

    public void removeReferencesFromParents() {
        Object parent = getParentObject();
        if (parent != null) {
            if (((mxCell)parent).getValue() instanceof EntityObject) {
                EntityObject entityObject = (EntityObject) ((mxCell)parent).getValue();
                entityObject.removeChildObject(this);
                entityObject.setSelfRelated(false);
            }
        }
    }
}

```



```

        } else if (((mxCell)parent).getValue() instanceof AssociativeEntityObject) {
            AssociativeEntityObject entityObject = (AssociativeEntityObject)
((mxCell)parent).getValue();
            entityObject.removeChildObject(this);
            entityObject.setSelfRelated(false);
        }
    }
}

public String getStyle() {
    return "relation";
}

public ConnectorObject getConnectorObject(mxCell relationCell, EntityObject entityObject) {
    ConnectorObject returnedObject = null;

    int count = relationCell.getEdgeCount();
    for (int i = 0; i < count; i++) {
        mxCell edge = (mxCell) relationCell.getEdgeAt(i);
        if (edge.getValue() instanceof ConnectorObject) {
            if ((edge.getSource() == relationCell &&
((mxCell)edge.getTarget()).getValue() instanceof EntityObject &&
(EntityObject)((mxCell)edge.getTarget()).getValue()
== entityObject) ||
            (edge.getTarget() == relationCell &&
((mxCell)edge.getSource()).getValue() instanceof EntityObject &&
(EntityObject)((mxCell)edge.getSource()).getValue()
== entityObject))
                returnedObject = (ConnectorObject)edge.getValue();
        }
    }

    return returnedObject;
}
}

```

```
package ufsc.sisinf.br.modelo2all.ui;
```

```
import java.awt.Color;
import java.awt.Component;
import java.awt.Graphics;
import java.awt.Insets;
import java.io.Serializable;
```

```
import javax.swing.border.Border;
```

```
/**
```

```
 * Border with a drop shadow.
```

```
 */
```

```
public class ShadowBorder implements Border, Serializable
```

```
{
```

```
    /**
```

```
     *
```

```
     */
```

```
    private static final long serialVersionUID = 6854989457150641240L;
```

```
    private Insets insets;
```

```
    public static ShadowBorder sharedInstance = new ShadowBorder();
```

```
    private ShadowBorder()
```

```

    {
        insets = new Insets(0, 0, 2, 2);
    }

    public Insets getBorderInsets(Component c)
    {
        return insets;
    }

    public boolean isBorderOpaque()
    {
        return false;
    }

    public void paintBorder(Component c, Graphics g, int x, int y, int w, int h)
    {
        // choose which colors we want to use
        Color bg = c.getBackground();

        if (c.getParent() != null)
        {
            bg = c.getParent().getBackground();
        }

        if (bg != null)
        {
            Color mid = bg.darker();
            Color edge = average(mid, bg);

            g.setColor(bg);
            g.drawLine(0, h - 2, w, h - 2);
            g.drawLine(0, h - 1, w, h - 1);
            g.drawLine(w - 2, 0, w - 2, h);
            g.drawLine(w - 1, 0, w - 1, h);

            // draw the drop-shadow
            g.setColor(mid);
            g.drawLine(1, h - 2, w - 2, h - 2);
            g.drawLine(w - 2, 1, w - 2, h - 2);

            g.setColor(edge);
            g.drawLine(2, h - 1, w - 2, h - 1);
            g.drawLine(w - 1, 2, w - 1, h - 2);
        }
    }

    private static Color average(Color c1, Color c2)
    {
        int red = c1.getRed() + (c2.getRed() - c1.getRed()) / 2;
        int green = c1.getGreen() + (c2.getGreen() - c1.getGreen()) / 2;
        int blue = c1.getBlue() + (c2.getBlue() - c1.getBlue()) / 2;
        return new Color(red, green, blue);
    }

    public static ShadowBorder getSharedInstance()
    {
        return sharedInstance;
    }
}

package ufsc.sisinf.brmodelo2all.model.shapes;

import com.mxgraph.swing.view.mxInteractiveCanvas;

```

```

public class ShapesCanvas extends mxInteractiveCanvas {

    /**
     * Static initializer.
     */
    static
    {
        putShape("attribute", new AttributeShape());
        putShape("identifierAttribute", new AttributeShape());
        putShape("inheritance", new InheritanceShape());
        putShape("column", new ColumnShape());
    }
}

package ufsc.sisinf.brmodelo2all.model.objects;

import java.util.Iterator;
import com.mxgraph.model.mxCell;
import com.mxgraph.util.mxResources;

import ufsc.sisinf.brmodelo2all.util.AppConstants;

public class TableObject extends ModelingObject {

    /**
     *
     */
    private static final long serialVersionUID = 6240155455942061860L;

    private int conversionOrder = -1;

    private final int NUMBER_OF_ATTRIBUTES = 3;

    public TableObject(String name) {
        super(name);
    }

    public void setConversionOrder(int conversionOrder) {
        this.conversionOrder = conversionOrder;
    }

    public int getConversionOrder() {
        return conversionOrder;
    }

    public int attributesCount(){
        return super.attributesCount() + NUMBER_OF_ATTRIBUTES;
    }

    public void getAttributes(int types[], String names[], String values[], boolean enabled[]) {
        super.getAttributes(types, names, values, enabled);

        int i = super.attributesCount();

        types[i] = -1;
        names[i] = null;
        enabled[i] = true;
        values[i++] = mxResources.get("fieldsNumber") + ": " +
Integer.toString(getChildObjects().size());
        types[i] = -1;
        names[i] = null;
        enabled[i] = true;

```

```

        values[i] = mxResources.get("keys") + ": [";
        values[i] += commaSeparatedPrimaryKeys();
        values[i++] += "]);
        types[i] = AppConstants.COMBO_BOX;
        names[i] = mxResources.get("conversionOrder");
        enabled[i] = true;
        values[i] = Integer.toString(conversionOrder);
    }

    public String commaSeparatedPrimaryKeys() {
        String result = "";
        Iterator<Object> iterator = getChildObjects().iterator();
        while (iterator.hasNext()) {
            Object object = iterator.next();
            ColumnObject columnObject = (ColumnObject)((mxCell)object).getValue();
            if (columnObject.isPrimaryKey()) {
                if (result != "")
                    result += ",";

                result += columnObject.getName();
            }
        }

        return result;
    }

    public String getToolTip() {
        String tip = "Tipo: Tabela<br>";

        tip += super.getToolTip();
        tip += mxResources.get("fieldsNumber") + ": ";
        tip += Integer.toString(getChildObjects().size());
        tip += "<br>";
        tip += mxResources.get("keys") + ": [";
        tip += commaSeparatedPrimaryKeys();
        tip += "]);
        tip += "<br>";

        return tip;
    }

    public int windowHeight() {
        return super.windowHeight() + 50;
    }

    public String[] getComboValues(String name){

        if (mxResources.get("conversionOrder") == name) {
            String[] values = { "Sem ordem" };
            return values;
        }

        return null;
    }

    public String getStyle() {
        return "table";
    }
}

package ufsc.sisinf.br.modelo2all.ui;

import javax.swing.BorderFactory;
import javax.swing.JToolBar;
import javax.swing.TransferHandler;

```

```

import ufsc.sisinf.brmodelo2all.ui.CommandActions.HistoryAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.NewConceptualModelingAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.NewLogicalModelingAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.OpenAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.PrintAction;
import ufsc.sisinf.brmodelo2all.ui.CommandActions.SaveAction;
//import com.mxgraph.swing.mxGraphComponent;
import com.mxgraph.swing.util.mxGraphActions;

public class ToolBar extends JToolBar
{
    /**
     *
     */
    private static final long serialVersionUID = -8015443128436394471L;
    // menu indexes
    public static final int NEW_CONCEPTUAL_BUTTON = 0;
    public static final int NEW_LOGICAL_BUTTON = 1;
    public static final int OPEN_BUTTON = 2;
    public static final int SAVE_BUTTON = 3;
    public static final int PRINT_BUTTON = 5;
    public static final int CUT_BUTTON = 7;
    public static final int COPY_BUTTON = 8;
    public static final int PASTE_BUTTON = 9;
    public static final int DELETE_BUTTON = 11;
    public static final int UNDO_BUTTON = 13;
    public static final int REDO_BUTTON = 14;

    /**
     *
     */
    public ToolBar(final AppMainWindow editor, int orientation)
    {
        super(orientation);
        setBorder(BorderFactory.createCompoundBorder(BorderFactory
            .createEmptyBorder(3, 3, 3, 3), getBorder()));
        setFloatable(false);
//        setPreferredSize(new Dimension(500, 48)); TODO apagar

        add(editor.bind("NewConceptual", new NewConceptualModelingAction(),
            "/ufsc/sisinf/brmodelo2all/ui/images/new_conceptual.png"));
        add(editor.bind("NewLogical", new NewLogicalModelingAction(),
            "/ufsc/sisinf/brmodelo2all/ui/images/new_logical.png"));
        add(editor.bind("Open", new OpenAction(),
            "/ufsc/sisinf/brmodelo2all/ui/images/open.png"));
        add(editor.bind("Save", new SaveAction(false),
            "/ufsc/sisinf/brmodelo2all/ui/images/save.png"));

        addSeparator();

        add(editor.bind("Print", new PrintAction(),
            "/ufsc/sisinf/brmodelo2all/ui/images/print.png"));

        addSeparator();

        add(editor.bind("Cut", TransferHandler.getCutAction(),
            "/ufsc/sisinf/brmodelo2all/ui/images/cut.png"));
        add(editor.bind("Copy", TransferHandler.getCopyAction(),
            "/ufsc/sisinf/brmodelo2all/ui/images/copy.png"));
        add(editor.bind("Paste", TransferHandler.getPasteAction(),
            "/ufsc/sisinf/brmodelo2all/ui/images/paste.png"));

        addSeparator();
    }
}

```

```

add(editor.bind("Delete", mxGraphActions.getDeleteAction(),
               "/ufsc/sisinf/brmodelo2all/ui/images/delete.png"));

addSeparator();

add(editor.bind("Undo", new HistoryAction(true),
               "/ufsc/sisinf/brmodelo2all/ui/images/undo.png"));
add(editor.bind("Redo", new HistoryAction(false),
               "/ufsc/sisinf/brmodelo2all/ui/images/redo.png"));

    }
}

```

7.2 Artigo

brModeloNext: a Nova Versão de uma Ferramenta para Modelagem de Bancos de Dados Relacionais

Leonardo Antonio Ramos, Otávio Soares Menna, Ronaldo dos Santos Mello

Depto. de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brasil.

{leoramos,otavio,ronaldo}@inf.ufsc.br

Resumo. *Este artigo apresenta a brModeloNext, uma ferramenta para modelagem de bancos de dados relacionais cujo principal diferencial é a produção de esquemas para as três etapas do projeto tradicional de um banco de dados: conceitual, lógica e física. Esta ferramenta é uma nova versão da brModelo, uma ferramenta popular utilizada no ensino de modelagem de dados em diversos cursos de Computação no Brasil. As principais melhorias desta nova versão são em aspectos de usabilidade, a portabilidade da ferramenta para uso em diversas plataformas e a correção de alguns erros.*

Abstract. *This paper presents brModeloNext, a tool for relational database modeling that novels by supporting the generation of schemata for all the three steps of the traditional database design methodology: conceptual, logical and physical. This tool is the new version of brModelo, a popular tool used by several Computer Science courses in Brazil for teaching of database modeling. The main improvements of brModeloNext are related to user interaction, portability, and the fixing of some bugs of the brModelo tool.*

1. Introdução

O projeto de um Banco de Dados (BD) é um processo importante, pois dele depende a geração de um repositório de dados a ser acessado por uma ou mais aplicações de uma organização, sendo estes dados vitais para o adequado funcionamento desta organização. Este processo visa garantir qualidade na definição dos dados, ou seja, garantir uma abstração adequada dos dados do domínio, bem como garantir armazenamento e acesso eficientes a estes dados [Battini et. al 1991]. A não consideração de uma metodologia de projeto de BD pode gerar diversos problemas, como inconsistência de dados, redundância, baixo desempenho e difícil manutenção.

A *brModelo*¹ é uma ferramenta livre e *open source* desenvolvida pelo Grupo de BD da UFSC² com a finalidade de apoiar a projeto de um BD relacional. Ela adota a metodologia de projeto tradicional de BD em três etapas (projetos conceitual, lógico e físico) e a notação do modelo Entidade-Relacionamento estendido (modelo EER) descritos em [Heuser, 2009]. Esta ferramenta tornou-se popular principalmente no meio acadêmico em atividades de ensino de modelagem de BDs relacionais, uma vez que possibilita, de forma amigável, a edição e a visualização de esquemas conceituais e

1 <http://www.sis4.com/brModelo/>

2 <http://www.gbd.inf.ufsc.br>

lógicos, um maior controle sobre o mapeamento conceitual-lógico, além da geração do esquema físico ou de implementação através do mapeamento do esquema lógico para um *script* SQL/DDL ANSI.

Apesar da aceitação pelos usuários, a *brModelo* apresenta algumas limitações relacionadas ao processo de modelagem conceitual e ao mapeamento entre esquemas conceitual e lógico. Em termos de modelagem conceitual, um exemplo é a impossibilidade de associar uma entidade genérica a mais de uma hierarquia disjunta de entidades especializadas. Em termos de mapeamento conceitual-lógico, a ferramenta alguns vezes posiciona incorretamente chaves estrangeiras e atributos de relacionamento no mapeamento de relacionamentos com certas cardinalidades. Além destas limitações, outro ponto fraco é a falta de portabilidade devido ao fato da *brModelo* ter sido desenvolvida em Delphi e executar apenas em plataformas Windows.

A *brModeloNext*³ é a nova versão da *brModelo* concebida com o intuito de sanar os problemas recém-citados. Ela foi desenvolvida com tecnologia Java, garantindo a portabilidade para qualquer plataforma, as limitações e erros foram corrigidos e aspectos de usabilidade foram aprimorados, visando torná-la mais intuitiva. Maiores detalhes sobre a ferramenta são apresentados na Seção 3. Além disso, a Seção 2 compara a *brModeloNext* com trabalhos relacionados e a Seção 4 é dedicada à Conclusão.

2. Trabalhos Relacionados

Diversas soluções comerciais e acadêmicas para projeto de BD relacional encontram-se disponíveis, como a *ERwin*⁴, *Rational Rose*⁵, *DBDesigner*⁶, *Enterprise Architect*⁷ e *TerraER* [Rocha and Terra, 2011]. Comparado com a maioria das ferramentas comerciais, o principal diferencial da *brModeloNext* é o suporte a todas as três etapas do projeto, com ênfase no apoio às modelagens conceitual e lógica. As ferramentas comerciais omitem a modelagem conceitual, iniciando o projeto a partir da modelagem das tabelas do BD. Uma exceção é a *Enterprise Architect*, que suporta modelagem conceitual EER. Entretanto, ela utiliza uma notação menos clara do EER e ocupa maior área de edição com atributos, da mesma forma que a *TerraER*. Outra desvantagem da *Enterprise Architect* em termos de usabilidade, se comparada à *brModeloNext*, é a falta de flexibilidade na alteração de conceitos, como por exemplo, promover um relacionamento a uma entidade associativa, ou um atributo a um atributo multivalorado. A *TerraER* também é uma ferramenta acadêmica, como a *brModeloNext*, porém ela suporta apenas a criação de modelagens conceituais.

Uma vantagem da *brModeloNext* em relação às soluções existentes é a sua flexibilidade na definição do mapeamento de esquemas conceituais para esquemas lógicos relacionais. Ela executa, de forma semi-automática, o processo de mapeamento, oferecendo ao usuário a possibilidade de escolher uma (1) dentre diversas alternativas de conversão de um determinado conceito da modelagem EER. Desta forma, o usuário tem a liberdade de orientar a conversão para uma estrutura lógica que seja mais

3 <http://www.inf.ufsc.br/~ronaldo/brModeloNext>

4 <http://www.erwin.com>

5 <http://www-01.ibm.com/software/rational/>

6 <http://www.fabforce.net/dbdesigner4/>

7 <http://www.sparxsystems.com.au/>

adequada para um determinado domínio. Esta e outras funcionalidades da ferramenta são detalhadas na próxima Seção.

3. Ferramenta *brModeloNext*: Implementação e Funcionalidades

A ferramenta *brModeloNext* suporta o desenvolvimento do projeto de um BD relacional desde a modelagem conceitual EER até a geração do esquema físico em SQL. Ela é uma evolução da ferramenta *brModelo*, pois foi re-implementada na linguagem Java, que oferece como vantagens a portabilidade de plataforma e um grande número de bibliotecas *open source* de qualidade. A semelhança entre a representação de esquemas de BDs e a estrutura de grafos motivou o uso do componente *JGraph*⁸ – uma biblioteca de renderização de grafos que foi a base da implementação da ferramenta.

A interface gráfica da *brModeloNext* é mostrada na Figura 1. Ela se baseia na interface da *brModelo*, propondo melhorias quanto à usabilidade. Uma barra de menu no topo da janela disponibiliza todas as funcionalidades da ferramenta seguida, logo abaixo, de uma barra de ferramentas com operações básicas de edição, como a gerência de arquivos e *undo/redo* de ações. Uma barra de objetos à direita da barra de ferramentas apresenta os objetos que podem ser incluídos na modelagem e possui sensibilidade ao contexto, ou seja, é exibida de acordo com o tipo de modelagem em edição no momento (conceitual ou lógica). Ainda, no rodapé, uma área para exibição de mensagens ao usuário (barra de status) provê o *feedback* das interações.

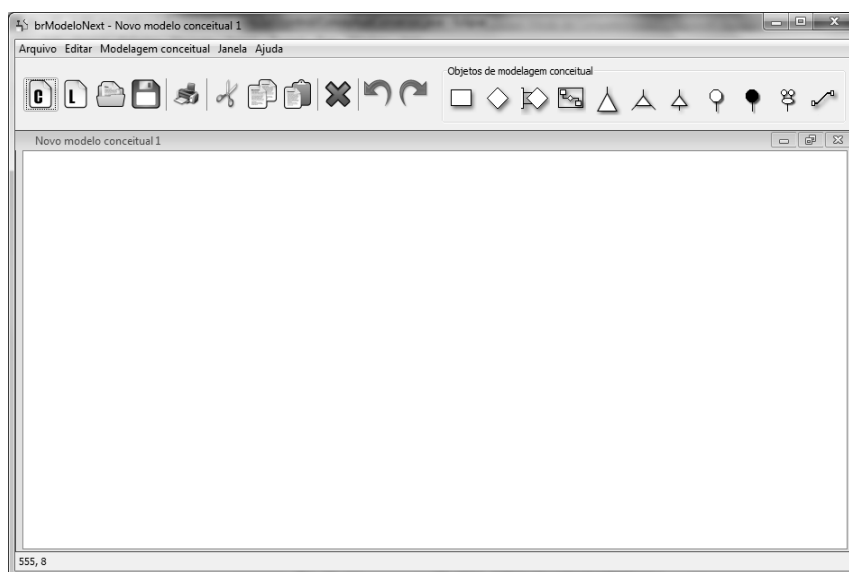


Figura 1. A interface gráfica da ferramenta *brModeloNext*

A barra de objetos de modelagem da Figura 1 ressalta o suporte da *brModeloNext* para os conceitos do EER. Além dos conceitos tradicionais do modelo ER, é possível a definição de relacionamentos de especialização disjuntos ou compartilhados e totais ou parciais, além da definição de atributos compostos e multivalorados, bem como de entidades associativas.

Uma importante melhoria quanto à usabilidade foi o aumento da área de edição das modelagens (centro da interface) através do reposicionamento da barra de objetos e

8 <http://www.jgraph.com>

da extinção do inspetor de objetos da *brModelo*. Em substituição ao inspetor de objetos, adotou-se a edição e consulta *in-place* de propriedades dos objetos, conforme ilustra a Figura 2. Esta edição é possível através de uma janela na qual o usuário edita as características pertinentes ao objeto inserido ou selecionado através de um duplo clique. As propriedades do objeto são visualizadas ao fixar o mouse sobre o mesmo.

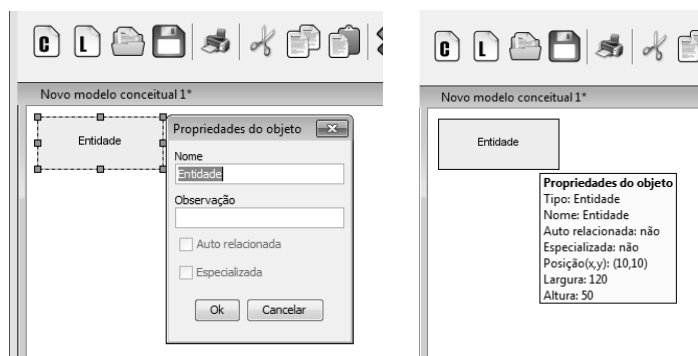


Figura 2. Exemplo de edição in-place ao inserir um objeto entidade

Outra melhoria foi a implementação do padrão de janelas MDI (*Multiple Documents Interface*) que possibilita a visualização de múltiplas modelagens ao mesmo tempo (Figura 3). Considerando que uma das principais tarefas da *brModeloNext* é a conversão de esquemas conceituais em esquemas lógicos, trabalhar simultaneamente com modelagens em diferentes níveis facilita a validação do projeto pelo usuário.

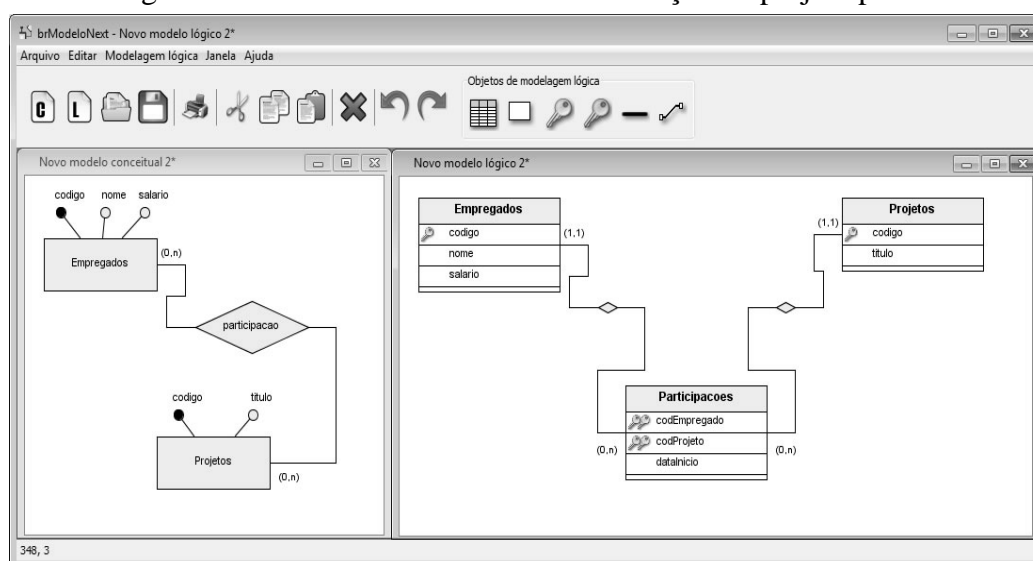


Figura 3. Visualização de múltiplas modelagens com o uso do padrão MDI

A Figura 3 também ilustra uma modelagem lógica relacional na janela da direita. A interface da *brModeloNext* facilita a visualização de relacionamentos entre tabelas através de chaves estrangeiras, sendo permitido ao usuário, mesmo após o mapeamento da modelagem conceitual, a validação do esquema lógico através da manipulação de tabelas (inserção, atualização ou remoção de atributos, alteração de nome, exclusão, etc). Esta flexibilidade garante que cada nível de modelagem seja manipulado livremente e de forma independente, possibilitando eventuais ajustes desejados pelo usuário para a melhoria da qualidade do projeto. Vale ressaltar que a ferramenta também salva modelagens em diferentes níveis em arquivos XML ou texto.

A ferramenta também controla restrições de modelagem durante o processo de projeto. Ela orienta, através de mensagens exibidas na barra de status, os passos que devem ser tomados na execução de uma atividade. Conforme visto na Figura 4, caso o usuário faça uma operação não permitida, a ferramenta emite um aviso descrevendo o erro. Neste exemplo, houve uma tentativa de inserir erroneamente uma especialização para um objeto do tipo relacionamento ao invés de um objeto do tipo entidade.

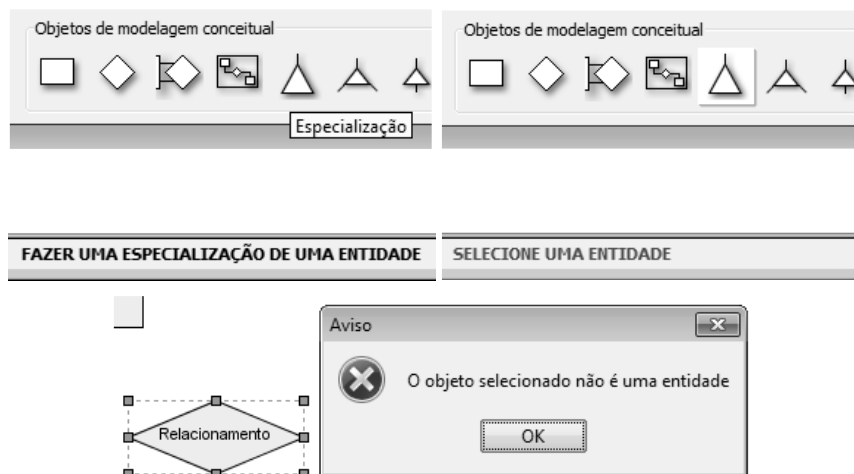


Figura 4. Exemplo de exibição de aviso de erro

Diferente de outras ferramentas similares, que executam o mapeamento conceitual-lógico de forma automática, a *brModeloNext* flexibiliza ao permitir que o usuário decida pela melhor alternativa de conversão em situações nas quais mais de uma alternativa possa ser empregada. Exemplos são o mapeamento de relacionamentos 1-1 ou 1-N e o mapeamento de hierarquias de especialização. Supondo um relacionamento opcional do tipo 1-N entre duas entidades *Automóveis* e *Pessoas*, no momento da conversão para o esquema lógico a ferramenta exibe uma janela para o usuário decidir entre duas possíveis alternativas (Figura 5): a geração de uma tabela para o relacionamento ou a geração de uma chave estrangeira na tabela *Automóveis*.

Uma vez finalizada uma modelagem lógica, a ferramenta permite a conclusão do projeto através da geração de um esquema físico (*script* SQL ANSI) com comandos de criação de tabelas e de restrições de chave estrangeira, como mostra a Figura 6.

3. Conclusão

Este artigo apresenta a nova versão de uma ferramenta destinada ao projeto de BDs relacionais denominada *brModeloNext*. A ferramenta não inova em termos de ser uma solução para uma problemática atual de gerência de dados, mas inova pelo conjunto de funcionalidades e aspectos de usabilidade, se comparada com soluções existentes. Tal fato justifica a sua utilização crescente no meio acadêmico.

As principais contribuições da ferramenta são o suporte a todas as etapas de modelagem da metodologia de projeto de BDs e a flexibilidade de manipulação de esquemas em qualquer nível de modelagem. Tal flexibilidade, inclusive na condução do mapeamento conceitual-lógico, facilita a compreensão do processo de projeto, tornando a ferramenta atrativa para o ensino de modelagem de dados em disciplinas de BD.

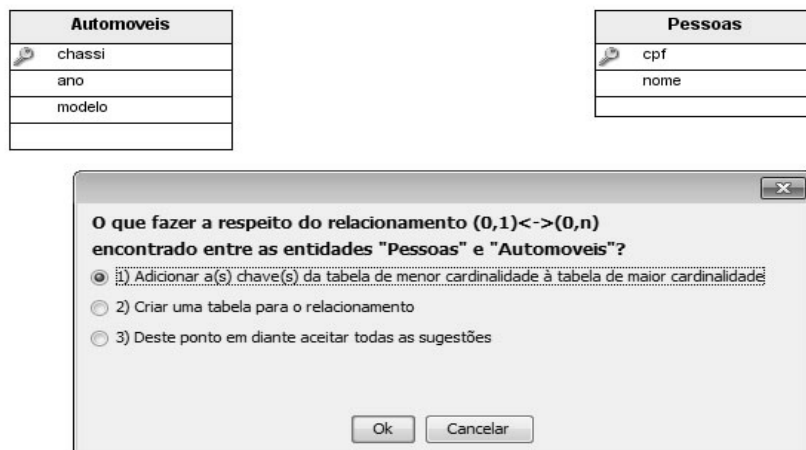


Figura 5. Interação com o usuário durante o projeto lógico

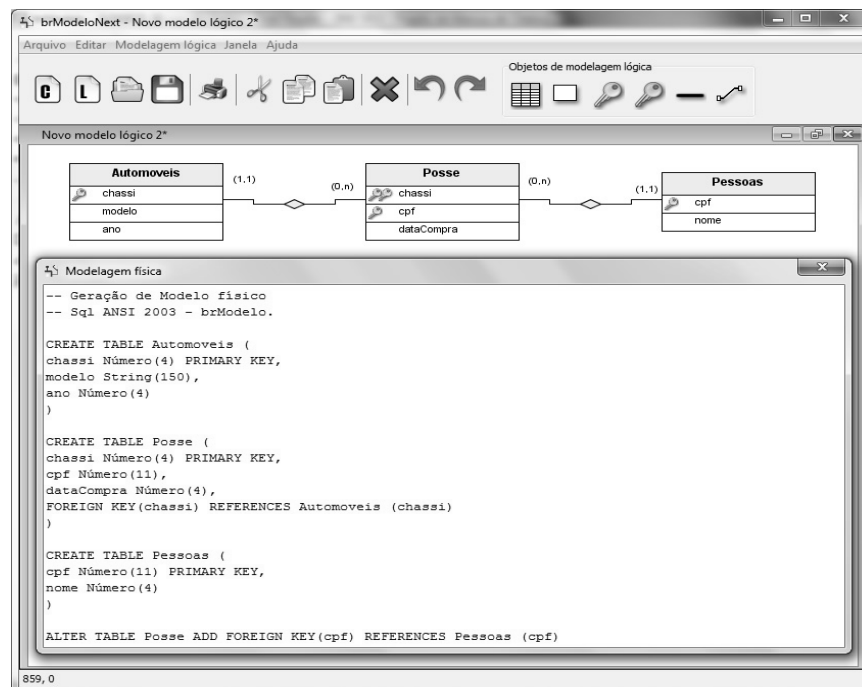


Figura 6. Exemplo de geração do *script* SQL/DDI para um projeto do BD

Trabalhos futuros incluem a realização de experimentos de usabilidade para validar a interface gráfica, a possibilidade de sincronização de alterações em um esquema para esquemas correspondentes em outros níveis de modelagem, bem como o suporte a uma metodologia de engenharia reversa de BDs relacionais.

Referências

- Batini, C.; Ceri, S.; Navathe, S. B. (1991), *Conceptual Database Design: An Entity-Relationship Approach*, Addison Wesley, 1st Edition.
- Heuser, C. A. (2009), *Projeto de Banco de Dados*, Bookman, 6^a Edição.
- Rocha, H. S. C. and Terra, R. (2010), "TerraER: Uma Ferramenta voltada ao Ensino do Modelo de Entidade-Relacionamento". In: VI Escola Regional de Banco de Dados (ERBD 2010), Joinville, SC.