UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIA DA COMPUTAÇÃO – BACHARELADO

FERRAMENTA WEB PARA MODELAGEM LÓGICA EM PROJETOS DE BANCOS DE DADOS RELACIONAIS

PAULO ALBERTO BUGMANN

BLUMENAU
2012

2012/2-22

PAULO ALBERTO BUGMANN

FERRAMENTA WEB PARA MODELAGEM LÓGICA EM PROJETOS DE BANCOS DE DADOS RELACIONAIS

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

Prof. Alexander Roberto Valdameri, Mestre – Orientador

BLUMENAU
2012

2012/2-22

FERRAMENTA WEB PARA MODELAGEM LÓGICA EM PROJETOS DE BANCOS DE DADOS RELACIONAIS

Por

PAULO ALBERTO BUGMANN

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

Presidente:	Prof. Alexander Roberto Valdameri, Mestre – Orientador, FURB
Membro:	Prof. Everaldo Artur Grahl, Mestre – FURB
Membro:	Prof. Jhony Alceu Pereira, Especialista – FURB

Blumenau, 10 de dezembro de 2012

Dedico este trabalho a minha namorada, minha família e amigos, que com seu apoio e compreensão me ajudaram na realização deste.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha namorada, Jéssica, por seu amor, compreensão e companheirismo nos momentos em que precisei.

Aos meus pais, por zelarem pela minha educação no ensino fundamental, médio e graduação.

Aos meus amigos, pelos empurrões e cobranças, em especial ao Fabio, pelas dicas na implementação web.

Ao meu orientador, Alexander Roberto Valdameri, por todo o auxílio concedido para a realização deste trabalho.

O único lugar onde o sucesso vem antes do trabalho é no dicionário.

Albert Einstein

RESUMO

Este trabalho apresenta a implementação de um aplicativo para modelagem lógica nos projetos de bancos de dados relacionais, desenvolvido para ser executado em ambiente web utilizando a plataforma Java, a arquitetura MVC e alguns padrões de projeto, como DAO e *Command*. Para tornar a interface do sistema mais amigável ao usuário, foi utilizado, em alguns momentos, o conjunto de técnicas conhecida por AJAX, disponibilizando recursos semelhantes aos de aplicativos *desktop*. O aplicativo desenvolvido foi elaborado a partir do trabalho WebModeler de Bachmann (2007), aperfeiçoando e incluindo novas funcionalidades. O aplicativo permite o controle de projetos de bancos de dados e suas versões, além de possibilitar a restrição do acesso dos usuários a determinados projetos. Também permite que sejam criados diagramas para modelar as tabelas, colunas e relacionamentos de forma gráfica, com a opção de imprimir, reduzir e ampliar o diagrama. Ainda, há a possibilidade cadastrar índices para as tabelas e gerar *scripts* com os comandos SQL necessários para criar o banco de dados modelado e também para atualizar um banco de dados já existente, aplicando as alterações efetuadas entre duas versões. Por fim, há a possibilidade de elaboração gráfica de diagramas UML como o diagrama de casos de uso e de classes e atividades.

Palavras-chave: Modelo lógico de dados. Bancos de dados relacionais. AJAX. Padrões de projeto. UML.

ABSTRACT

This study presents the implementation of an application for modeling logic in the design of relational databases, developed to run in a web environment using the Java platform, the MVC architecture and some design patterns such as DAO and Command. To make the interface more user-friendly system was used at times, the set of techniques known as AJAX, offering features similar to desktop applications. The application developed was drawn from the work of WebModeler Bachmann (2007), including new features and improving. The App allows control of projects databases and their versions, and enables the restriction of user access to certain projects. It also allows diagrams to be created to model the tables, columns and relationships graphically, with the option of printing, reduce and enlarge the diagram. Still, there is the possibility to register the tables and indexes scripts to generate the SQL statements needed to create the database modeled and also to upgrade an existing database, applying the changes made between two versions. Finally, there is the possibility of developing graphical UML diagrams such as use case diagram and classes and activities.

Key-words: Logical data model. Relational databases. AJAX. Design patterns. UML.

LISTA DE ILUSTRAÇÕES

LISTA DE SIGLAS

AJAX – Asynchronous Javascript and XML

BD – Banco de Dados

CGI – Common Gateway Interface

CSS – Cascading Style Sheets

DOM – Document Object Model

HTML - Hypertext Markup Language

HTTP – Hypertext Transfer Protocol

IIS – Internet Information Services

IDE – Integrated Development Environment

JSP – Java Server Pages

MVC – Model View Controller

SGDB – Sistemas Gerenciadores de Banco de Dados

SQL – Structured Query Language

XHTML – Extensible Hypertext Markup Language

XML – Extensible Markup Language

XSL – Extensible Style Language

UML – Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

Atualmente os sistemas de computadores automatizam uma grande parte do trabalho, possibilitando assim, um ganho de produção para organizações de diversos setores. Geralmente um sistema informatizado manipula um conjunto de dados. Estes dados por sua vez podem ser compartilhados entre várias áreas de atuação nas organizações. Segundo Heuser (2000, p. 4), para manter grandes repositórios compartilhados de dados, ou seja, para manter banco de dados, são usados Sistemas de Gerenciamento de Banco de Dados (SGBD). No mercado de software há um grande predomínio do SGBD do tipo relacional. Esta realidade faz com que seja necessário dar um enfoque cada vez mais expressivo no processo de projeto de banco de dados relacionais, para garantir que a estrutura criada atenda requisitos como desempenho aceitável na recuperação dos dados, por exemplo.

Heuser (2000, p. 9) divide o projeto de banco de dados, basicamente, em duas fases:

a) modelagem conceitual, onde é construído um diagrama que captura as necessidades da organização em termos de armazenamento de dados, sem a preocupação com a forma como esta estrutura será implementada;
b) projeto lógico, que transforma o modelo conceitual em um modelo lógico, definindo como o banco de dados será implementado em um tipo específico de SGBD.

Cougo (1997, p. 157) enfatiza que muitos autores que abordam o tema da modelagem de dados, direcionam o processo diretamente para o nível lógico. Muitas empresas desenvolvedoras de software seguem o mesmo princípio, ou seja, não executam a etapa de modelagem conceitual e, através de ferramentas do tipo *Computer Aided Software Engineering* (CASE), implementam apenas o modelo lógico.

No mercado há uma vasta variedade de ferramentas CASE com o propósito de auxiliar na modelagem de bancos de dados relacionais, a maioria destas ferramentas têm características em comum, como a possibilidade de modelar os dados de forma gráfica (desenhando tabelas e relacionamentos) e salvar os modelos criados em arquivos binários, para serem enviados a outro computador ou pessoa que vá utilizar este modelo, desde que estas utilizem ferramentas compatíveis. Outra característica marcante presente nestes softwares é a necessidade de instalá-los no computador de quem for utiliza-lo, isto é, são aplicações *desktop*, sendo que algumas necessitam de licença para serem executadas e outras

são consideradas de código aberto, podendo ser instaladas e utilizadas sem a necessidade de pagamento.

Além do custo para aquisição, instalação e manutenção, as aplicações *desktop* tem outra grande desvantagem, na atualização do software este precisa ser feita em cada computador que tenha o software instalado, o que ocasionar diferentes versões no mesmo grupo de trabalho e até demandar um alto tempo para esta total atualização.

Uma alternativa interessante para amenizar eventuais problemas decorrentes destas características dos aplicativos *desktop* é a aplicações web, que vem passando por transformações significativas nos últimos anos. Neste momento, a web está virando uma plataforma: todo tipo de aplicativo é executado no *browser* (FORTES, 2006, p. 44). Inicialmente, foram os serviços de e-mail que saíram do *desktop* e foram disponibilizados na grande rede mundial de computadores. Atualmente, há editores de textos, planilhas, agendas, diários, álbuns de fotos, comércios eletrônicos, enciclopédias e muitos outros serviços como estes. Além disso, diversos softwares corporativos são executados nos navegadores, como os portais de compras e do governo, por exemplo. Kurniawan (2002, p. XIV), ratifica esta situação, afirmando que um *browser* não é mais utilizado apenas para exibir páginas estáticas na internet, pois é muito comum vê-lo sendo utilizado como um cliente de um aplicativo.

Diante deste contexto, pode-se dizer que há uma grande tendência de transformar os aplicativos *desktop* em ferramentas para o ambiente web e os softwares de modelagem de banco de dados também podem ser enquadrados nesta situação.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é aprimorar e incluir novas funcionalidades ao trabalho "Aplicativo web para definição do modelo lógico no projeto de banco de dados relacional" de Bachmann (2007).

Os objetivos específicos do trabalho são:

a) disponibilizar um recurso de impressão de diagramas;
b) disponibilizar uma aplicação de *zoom* na visualização e possibilidade de personalizar os relacionamentos, alterando suas posições e o seu formato;
c) alterar o módulo de geração de *scripts* para permitir a geração de arquivos compatíveis com outros SGBD relacionais;

d) disponibilizar uma rotina de engenharia reversa, a qual se conecta a um banco de dados e importa a sua estrutura, gerando um diagrama inicial;
e) disponibilizar o cadastro de índices nas tabelas do banco de dados;
f) disponibilizar a funcionalidade para implementação de alguns diagramas da UML, como diagrama de caso de uso, diagrama de classe e de atividades.

1.2 ESTRUTURA DO TRABALHO

Este trabalho está organizado em quatro capítulos.

O primeiro é onde se tem a introdução, a justificativa do trabalho, o objetivo geral e objetivos específicos.

No segundo, há a fundamentação teórica do trabalho, discorrendo a cerca de temas como projeto de banco de dados e bancos de dados relacionais, além de apresentar as principais tecnologias utilizadas no desenvolvimento do sistema, bem como trabalhos correlatos ao presente.

O terceiro capítulo descreve o processo de análise e implementação do aplicativo, sendo ilustrado por diagramas que visam facilitar a compreensão do mesmo. Também contem um exemplo da operacionalidade do aplicativo, onde a modelagem de um projeto de banco de dados relacional é demonstrada passo-a-passo no sistema.

Por fim, há o quarto capítulo, onde constam a conclusão e sugestões de extensões deste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo visa apresentar conceitos, métodos, técnicas e tecnologias envolvidas na realização do trabalho.

As primeiras seções trazem os conceitos básicos relacionados à área de banco de dados, como projeto de banco de dados e bancos de dados relacionais. As seções seguintes abordam as técnicas e tecnologias utilizadas na implementação da ferramenta, como AJAX, por exemplo. Por fim, há a seção que versa sobre trabalhos correlatos, relacionando suas características mais importantes e as semelhanças existentes entre eles e o presente trabalho.

2.1 SISTEMA GERENCIADOR DE BANCO DE DADOS (SGBD)

Um SGBD é, conforme simplifica Date (2000, p. 37), um software que trata de todo o acesso ao banco de dados. No mesmo sentido, Heuser (2000, p. 5) descreve o SGBD como um software que incorpora as funções de definição, recuperação e alteração de dados em um banco de dados, sendo que além destas funcionalidades, Date (2000, p. 38) cita a segurança e a integridade dos dados como fatores de responsabilidade do SGBD. Heuser (2000, p. 83) define que um dos objetivos primordiais de um SGBD é a integridade de dados. Para dizer que os dados estão íntegros significa dizer que eles refletem corretamente a realidade representada pelo banco de dados e que são consistentes entre si.

Há muitos tipos diferentes de SGBD. Desde pequenos sistemas que funcionam em computadores pessoais a sistemas enormes que estão associados a *mainframes*, dentre os quais pode-se citar o relacional, o hierárquico, o de rede e o orientado a objetos, cada qual possuindo características diferenciadas na implementação das funcionalidades descritas anteriormente. O foco deste trabalho está no modelo relacional.

2.2 BANCO DE DADOS RELACIONAL

Date (2000, p. 67) descreve um banco de dados relacional como sendo uma coleção de

tabelas, também chamadas de relações. Estas tabelas são formadas por colunas (ou atributos), sendo que cada uma delas pode receber um valor pertencente a um domínio. Heuser (2000, p. 83) define domínio como o conjunto de valores que podem ser alfanumérico, numérico, entre outros, que devem ser especificados para cada coluna de tabela. Cada conjunto de atributos forma uma linha, também chamada de registro. Uma tabela pode possuir inúmeros registros.

Cougo (1997, p. 162) cita outro conceito básico relativo aos bancos de dados relacionais: o conceito de chave primária. A chave primária garante que não existirão dois registros iguais dentro da mesma tabela, sendo formada por atributos que, por análise prévia ou conhecimento do ambiente que está sendo modelado, sabe-se que não terão valores repetidos em dois ou mais registros.

O conceito de chave estrangeira também é fundamental no banco de dados relacional, pois é através dela que os relacionamentos entre tabelas são estabelecidos (COUGO, 1997, p. 166). A chave estrangeira nada mais é do que a chave primária (conjunto de atributos únicos) de uma tabela que é repetida em outra tabela, sendo que os valores da chave na segunda tabela devem existir na tabela original para garantir a integridade das informações.

2.3 PROJETO DE BANCO DE DADOS

Segundo Date (2000, p. 287), o projeto de banco de dados é o processo através do qual se define uma estrutura lógica adequada para armazenar algum conjunto de dados que precisa ser representado num banco de dados. Heuser (2000, p. XIII) afirma que este processo pode ocorrer em três etapas. A primeira é chamada de modelagem conceitual, que procura capturar formalmente quais as informações que precisarão ser gravadas num banco de dados.

A segunda etapa é o projeto lógico, que define as estruturas de dados necessárias para atender os requisitos identificados na modelagem conceitual, sendo que estas estruturas dependerão diretamente do tipo de SGBD utilizado.

Finalmente, a terceira etapa é o projeto físico, que define os parâmetros físicos de acesso ao banco de dados, procurando aperfeiçoar a desempenho do sistema como um todo.

No projeto de banco de dados, há uma funcionalidade de engenharia reversa, que é o processo para auxiliar na recuperação da estrutura, de um banco de dados, que por algum motivo não tenha o modelo definido. Isto facilita o coomprendimento para futuros projetos em bancos de dados que não se tem documentação sobre a sua estrutura e modelo lógico.

2.3.1 Modelo lógica de dados

De acordo com Heuser (2000, p. 7), “um modelo lógico é a descrição de um banco de dados no nível de abstração visto pelo usuário do SGBD”. Cougo (1997, p. 29) segue a mesma linha, que defini o modelo lógico de dados sendo aquele em que os objetos, características e relacionamentos tem a representação de acordo com regras de implementação e limitações impostas por algum tipo de tecnologia, isso é, algum tipo de SGBD.

No caso dos SGBD do tipo relacional, o modelo lógico “deve definir quais as tabelas que o banco contém e, para cada tabela, quais os nomes das colunas” (HEUSER, 2000, p. 7). Cougo (1997, p. 29) acrescenta ainda que este modelo deve ser elaborado respeitando conceitos como chaves de acesso, controles de chaves duplicadas, itens de repetição, normalização, ponteiros, *headers*, integridade referencial entre outros.

2.4 FERRAMENTA *COMPUTER-AIDED SOFTWARE ENGINEERING* (CASE)

A engenharia de software abrange um conjunto de três elementos fundamentais que são, métodos, ferramentas e procedimentos que possibilita ao gerente o controle do processo de desenvolvimento do software e oferece ao profissional uma base para a construção de software de alta qualidade e produtivamente, segundo PRESSMAN (1995, p. 31). As ferramentas de engenharia de software proporcionam apoio automatizado ou semiautomatizado aos métodos, sendo assim denominadas de Ferramentas *Computer-Aided Software Engineering* (CASE).

As ferramentas CASE podem ser classificadas por função, por seus papéis como instrumentos para os gerentes e para o pessoal técnico, pelo uso que elas têm nas várias etapas do processo de engenharia de software, pela arquitetura do ambiente (hardware ou software) que as suporta ou até mesmo pela origem ou custo.

Podem ser consideradas como ferramentas automatizadas que tem como objetivo auxiliar o desenvolvedor de sistemas em uma ou várias etapas do ciclo de desenvolvimento de software, inclusive na modelagem do banco de dados.

2.5 AJAX

O AJAX, sigla de Asynchronous Javascript And XML permite o desenvolvimento de aplicativos web com interface “rica”. Crane e Pascarello (2005, p. 5) definem que “rica” refere-se ao modelo de interação do aplicativo com o usuário. Um modelo rico de interação com o usuário é aquele que suporta uma variedade de formas de entrada e responde intuitivamente e em tempo hábil a estas entradas.

Segundo Garrett (2005), o AJAX não é uma tecnologia. Na realidade são várias tecnologias que juntas se combinam de uma maneira extremamente poderosa. O AJAX incorpora:

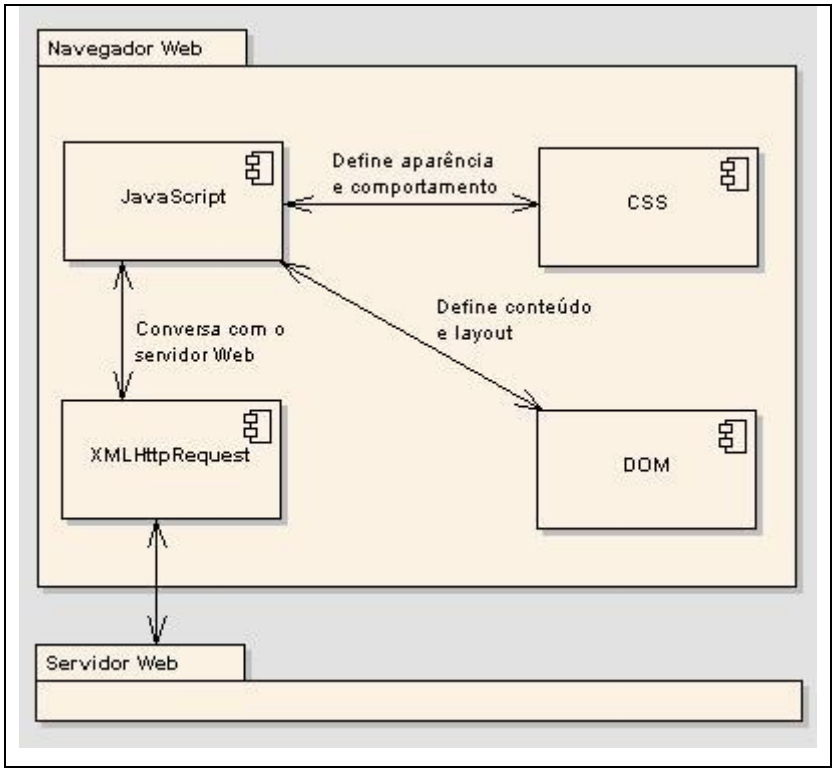
a) Cascading Style Sheets (CSS) e XHTML para exibição das informações;

b) Document Object Model (DOM) para interagir dinamicamente com as informações apresentadas;

c) XMLHttpRequest para trocar dados de forma assíncrona com o servidor;

d) Javascript como linguagem de script para suporte às tecnologias acima.

O relacionamento entre estes quatro elementos numa aplicação AJAX pode ser visto na figura 1.



Fonte: adaptado de Crane, Pascarello e James (2007, p. 23).

Figura 1 – Relacionamento entre os componentes do AJAX

Estas tecnologias são tratadas pelo *browser*, ou seja, "o AJAX é uma abordagem do lado cliente e pode interagir com a JEE, .NET, PHP, Ruby e scripts CGI – realmente não depende do servidor" (ASLESON;SCHUTTA, 2006, p. 13).

O detalhe mais importante dessa interação é que, ao contrário dos aplicativos web comuns, após a solicitação ser feita ao servidor não ocorre à recarga total da página, o que melhora em muito a usabilidade do sistema, sendo este um dos benefícios da utilização do AJAX para obtenção de uma interface rica para as aplicações web.

2.6 TRABALHOS CORRELATOS

Existem disponíveis ferramentas comerciais e acadêmicas que abordam o tema modelagem de banco de dados. Desta maneira, é possível encontrar desde trabalhos acadêmicos até ferramentas comerciais. Dentre este foram selecionados dois trabalhos acadêmicos, os quais são:

a) Bernardi (1997) desenvolveu uma ferramenta que possibilita a modelagem de um banco de dados relacional, cadastrando as tabelas com suas colunas e relacionamentos;
b) Cândido (2007) desenvolveu uma ferramenta *freeware* voltada para ensino de modelagem em banco de dados relacional com base na metodologia defendida por Heuser (2000).

2.6.1 Protótipo de uma ferramenta de Entidade Relacionamento (ER) para geração de código para banco de dados Interbase

A ferramenta proposta por Bernardi (1997) possibilita a modelagem de um banco de dados relacional, cadastrando as tabelas com suas colunas e relacionamentos, sendo um aplicativo *desktop* cujo principal objetivo era, além do cadastro da estrutura do banco, a geração de um arquivo-texto (*script*) com os comandos na linguagem *Structured Query Language* (SQL) necessários para criar o banco em um SGBD específico, o Interbase.

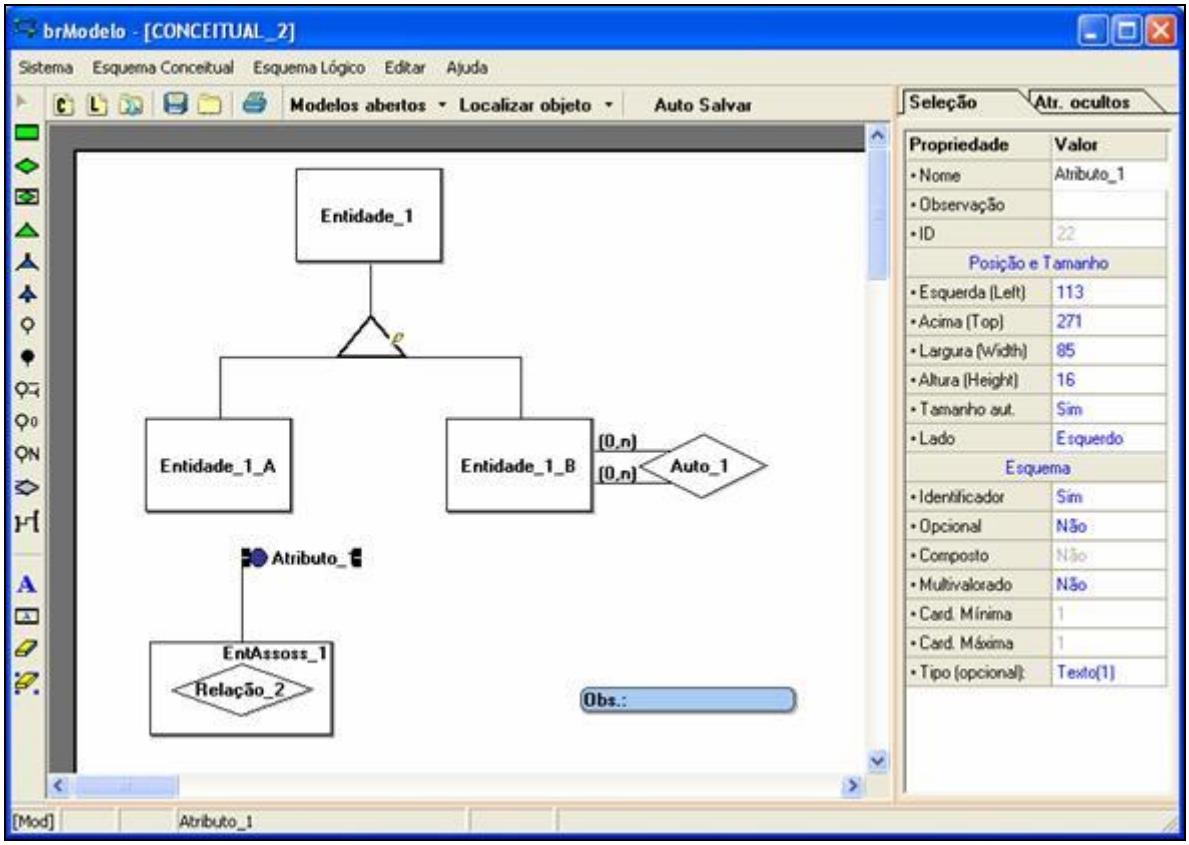
A linguagem de programação utilizada para a criação deste protótipo é o Borland

Delphi versão 3 e o sistema operacional utilizado foi o Microsoft Windows 95.

2.6.2 brModelo: ferramenta de modelagem conceitual de banco de dados

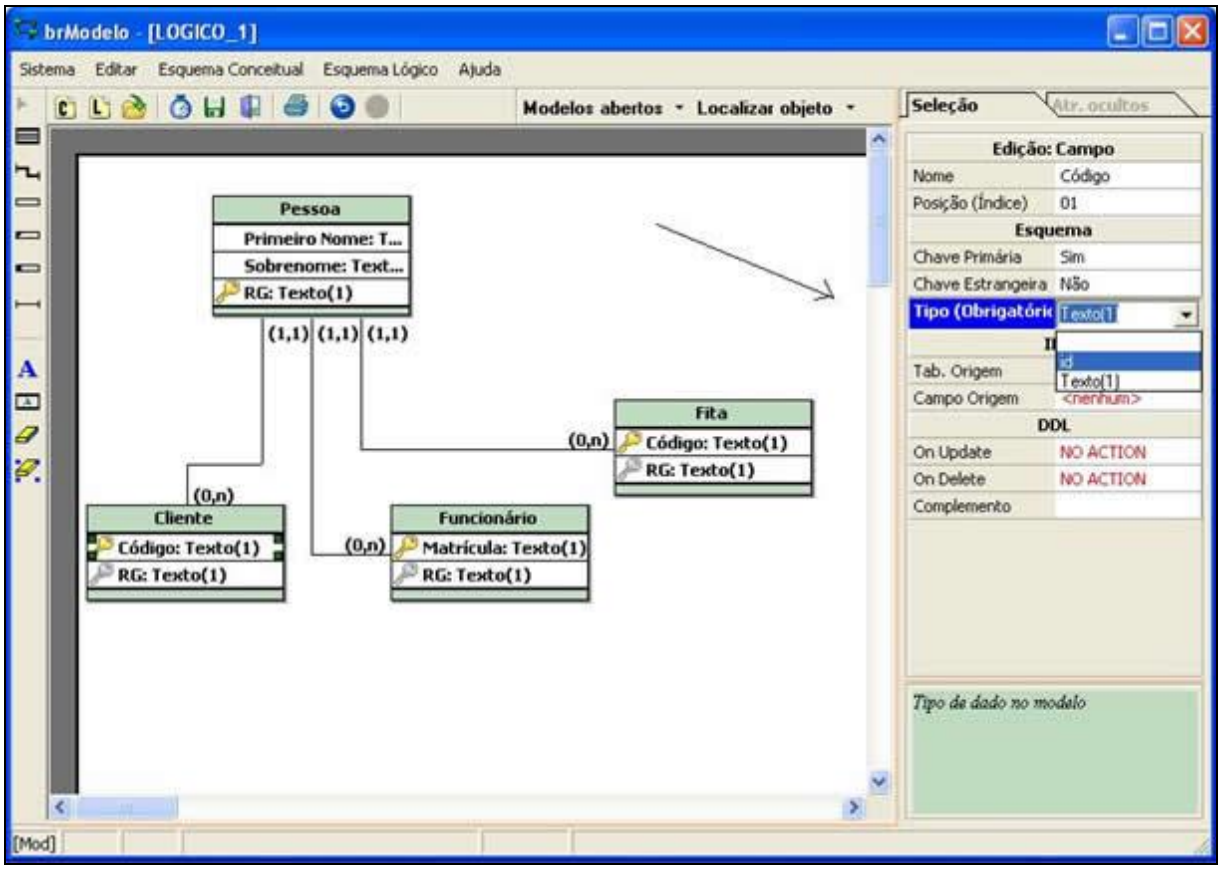
A ferramenta proposta por Cândido (2007) é voltada para o ensino de modelagem de dados em banco relacional que implemente exatamente os conceitos de criação de modelos de uma forma didática, simples, clara e de fácil assimilação de forma independente do SGBD adotado.

A figura 2 e 3 mostram a tela do sistema brModelo, com um exemplo de utilização.



Fonte: Cândido (2007).

Figura 2 - Implementa os diagramas com sutis diferenças se comparada a notação original



Fonte: Cândido (2007).

Figura 3 - Modelo lógico: modelo com template de conversão

2.7 APLICAÇÃO WEB MODELER

A ferramenta proposta por Bachmann (2007) possibilita a definição do modelo lógico nos projetos de bancos de dados relacionais, desenvolvida para o ambiente web. Para tornar a interface do sistema mais amigável ao usuário foi utilizado, em alguns momentos onde era necessário recursos gráficos mais elaborados, o conjunto de técnicas conhecida por *Asynchronous Javascript and XML (*AJAX), disponibilizando recursos semelhantes aos de aplicativos *desktop*. O aplicativo permite o controle de projetos de bancos de dados e suas versões, além de possibilitar a restrição do acesso dos usuários a determinados projetos. Também permite que sejam criados diagramas para modelar as tabelas, colunas e relacionamentos de forma gráfica. Ainda há a possibilidade de gerar *scripts* com os comandos SQL necessários para criar o banco de dados modelado e também para atualizar um banco de dados já existente, aplicando as alterações efetuadas entre duas versões.

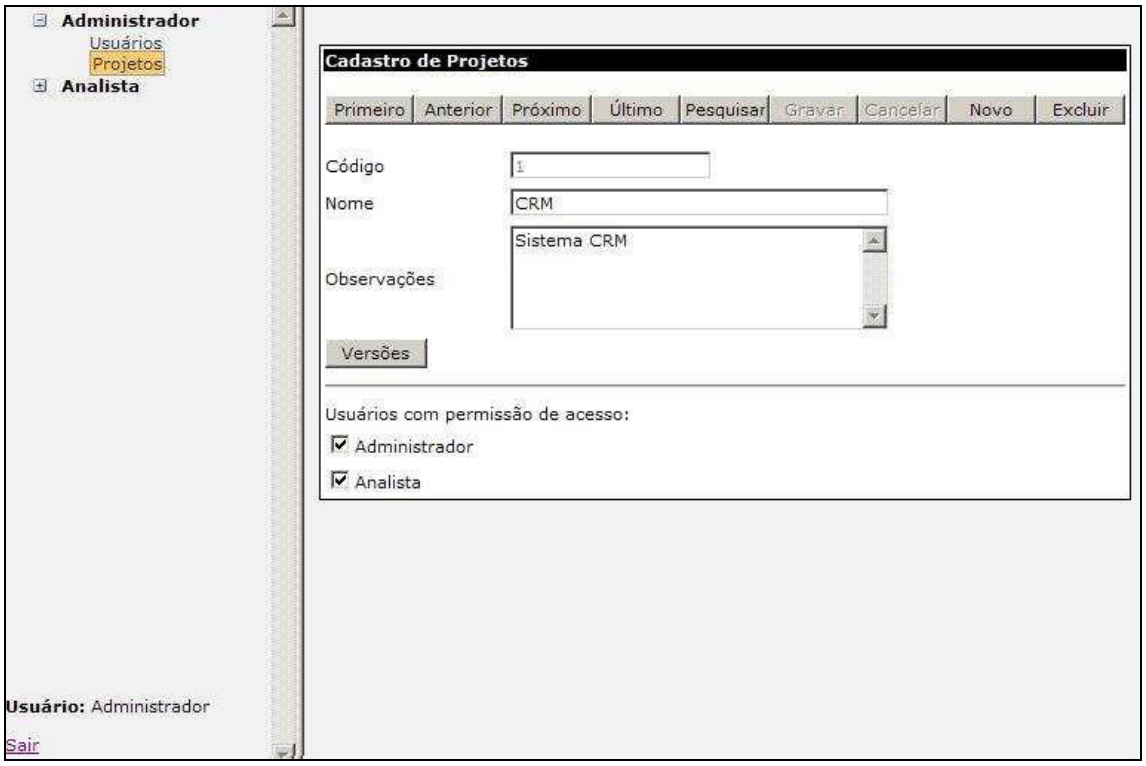
A linguagem de programação utilizada para a criação do "Aplicativo Web para

Definição do Modelo Lógico" é Java, a arquitetura *Model View Controller* (MVC) e alguns padrões de projeto, como *Data Access Object* (DAO) e *Command.*

As figuras 4, 5 e 6 mostram telas de cadastro do sistema.

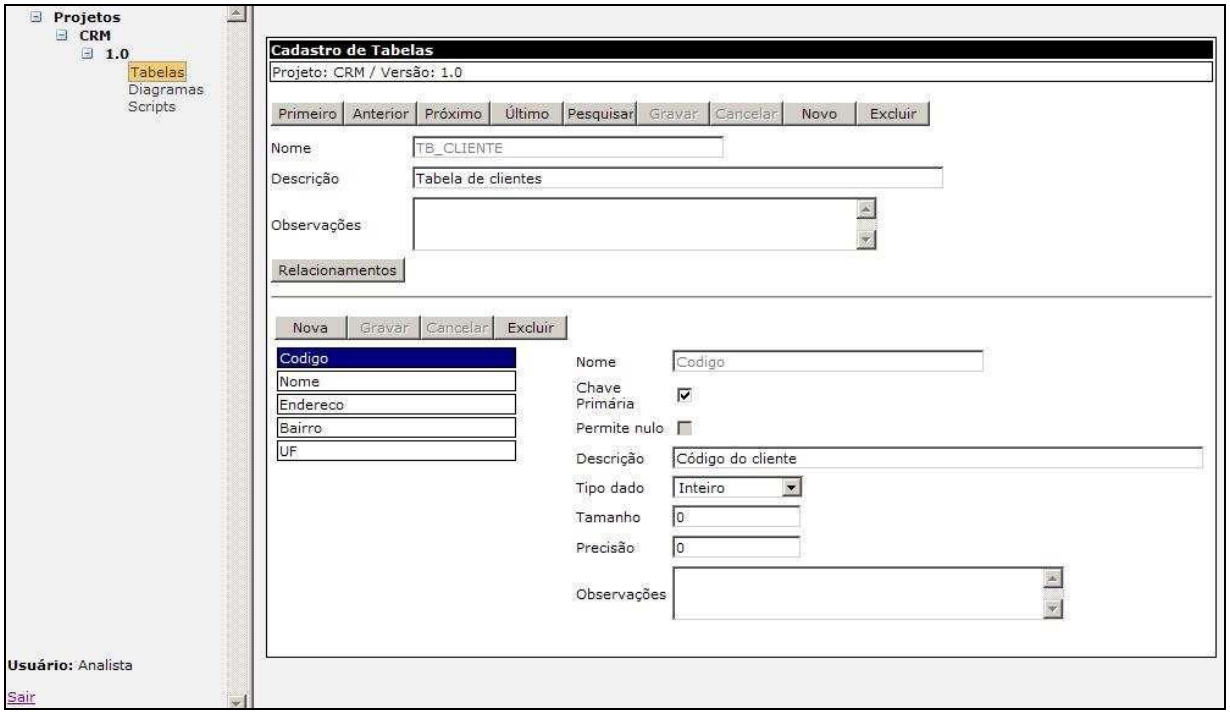
Fonte: Bachmann (2007, p. 74).

Figura 4 – Tela principal do sistema



Fonte: Bachmann (2007, p. 76).

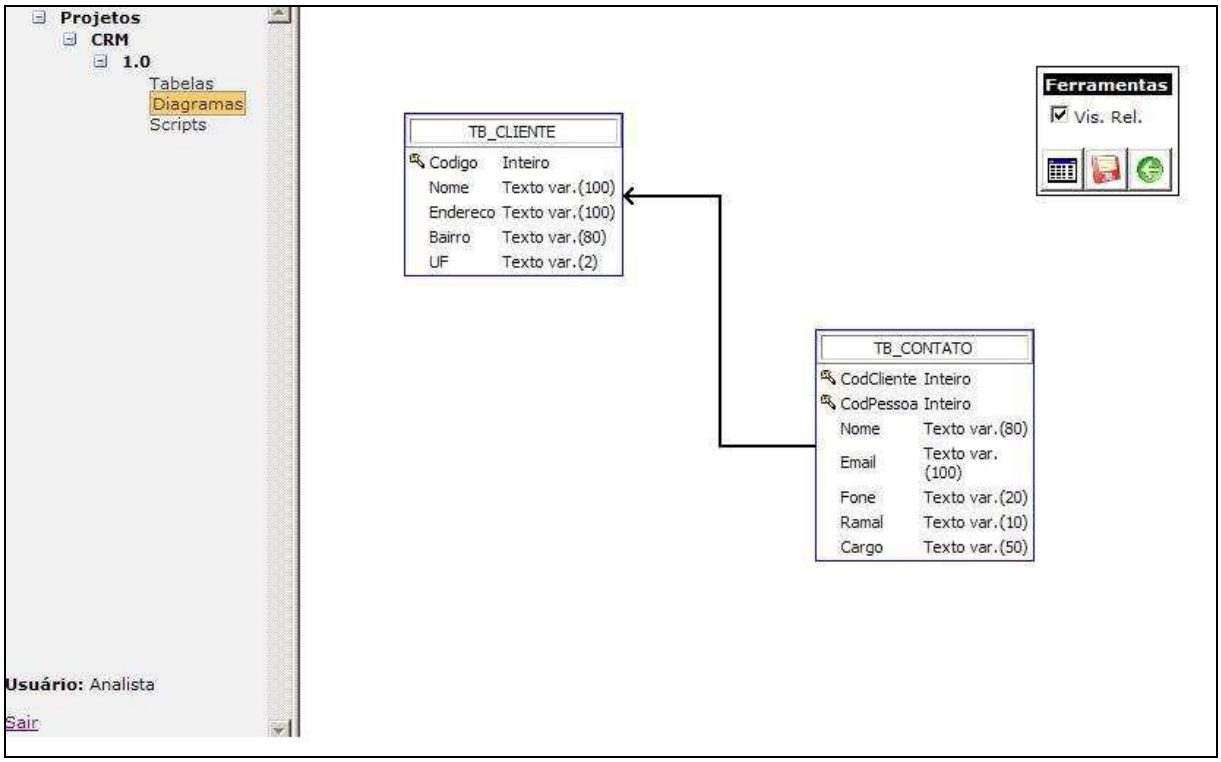
Figura 5 - Tela de cadastro de projetos



Fonte: Bachmann (2007, p. 78).

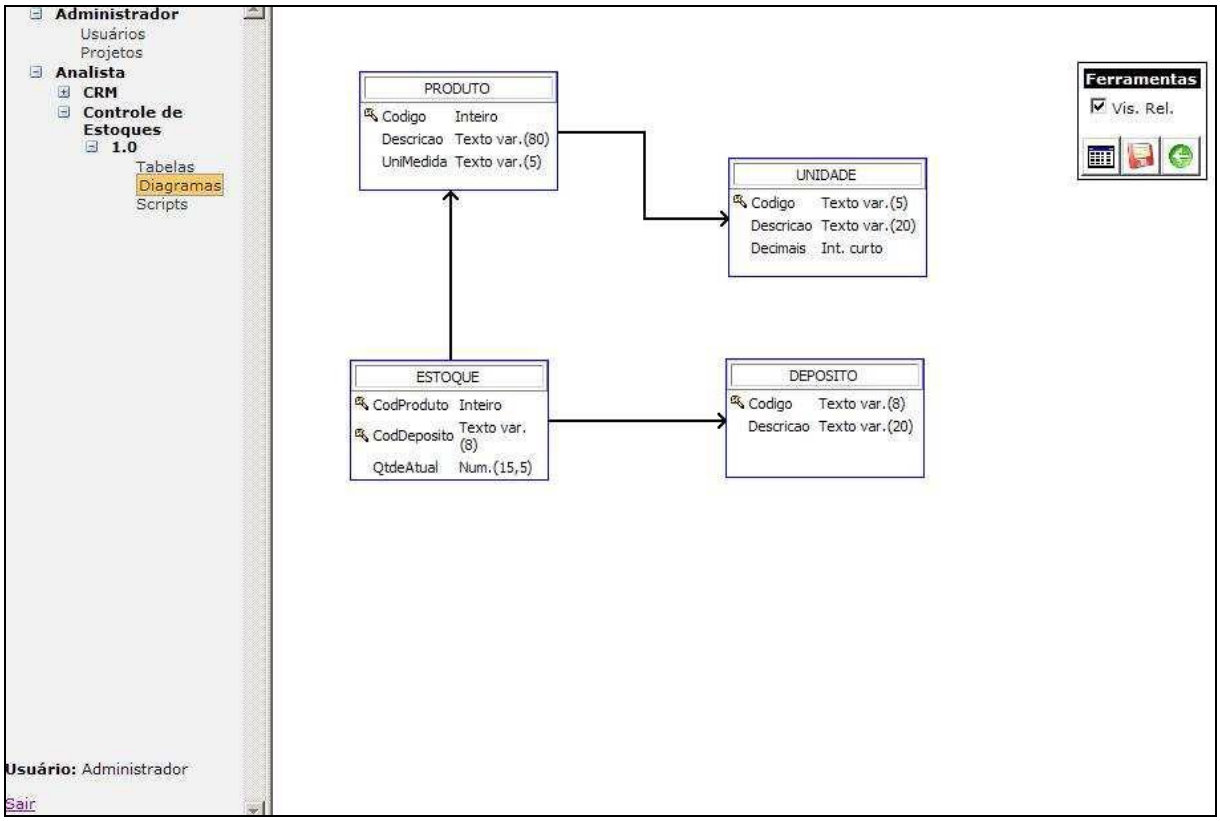
Figura 6 - Tela de cadastro de tabelas

As figuras 7 e 8 mostram a modelagem feita pelo aplicativo de Bachmann (2007).



Fonte: Bachmann (2007, p. 82).

Figura 7 - Relacionamento entre tabelas na modelagem gráfica



Fonte: Bachmann (2007, p. 85).

Figura 8 - Modelagem no WebModeler

3 DESENVOLVIMENTO

Este capítulo descreve o processo de desenvolvimento do aplicativo, apresentando a análise dos requisitos, a especificação do sistema e a sua implementação. Além disso, mostra como o AJAX e padrões de projeto foram utilizados em conjunto com a especificação JEE.

As etapas do desenvolvimento do software foram:

a) definição dos requisitos: as principais funcionalidades necessárias ao sistema foram identificadas e documentadas;
b) especificação do sistema: foram criados os diagramas da *Unified Modeling Language* (UML) pertinentes ao sistema, além da definição do modelo do banco de dados da aplicação;
c) implementação: com os requisitos definidos e com o término da especificação do sistema, ocorreu o desenvolvimento do mesmo, com testes sendo executados conjuntamente.

3.1 REQUISITOS PRINCIPAIS DA FERRAMENTA

Os softwares de modelagem de banco de dados já existentes no mercado, como o PowerDesigner da Sybase, por exemplo, serviram como fonte para obtenção de alguns requisitos funcionais. A estes, foram adicionados outros por Bachmann (2007), como o controle de versões de projetos e a restrição de acesso dos usuários aos projetos e dando continuidade ao trabalho foram melhorados os itens, como aproximar, distanciar e imprimir diagramas e ainda a geração de scripts de para criação e atualização de demais bancos de dados relacionais disponíveis no mercado.

Foram adicionadas novas funcionalidades, como o cadastro de índices de tabelas, cadastro de regras de negócio e geração de diagramas UML. O quadro 1 apresenta os requisitos funcionais do sistema.

REQUISITOS FUNCIONAIS
RF001: O sistema deverá permitir o cadastramento dos usuários que poderão acessá-lo.
RF002: O sistema deverá possibilitar o cadastramento dos projetos controlados.
RF003: O sistema deverá possibilitar a ligação entre os projetos e os usuários que terão permissão de acesso aos mesmos.
RF004: O sistema deverá permitir o controle das versões de cada projeto.
RF005: O sistema deverá permitir o cadastramento das tabelas existentes em cada versão, assim como suas colunas e relacionamentos.
RF006: O sistema deverá possibilitar a geração de arquivos-texto com os comandos SQL para a criação do banco de dados nos SGBD Oracle, MSSqlServer, MySql, Firebirb e PostgreSql.
RF007: O sistema deverá permitir a geração de arquivos-texto com os comandos SQL para a atualização do banco de dados nos SGBD Oracle, MSSqlServer, MySql, Firebirb e PostgreSql, baseado na comparação das diferenças entre duas versões do projeto.
RF008: O sistema deverá emitir um relatório com a descrição das diferenças existentes entre uma versão e outra do projeto, identificando tabelas e colunas que tenham sido incluídas, alteradas ou excluídas.
RF009: O sistema deverá permitir a impressão de diagramas.
RF010: O sistema deverá possibilitar a aproximação e redução dos diagramas.
RF011: O sistema deverá permitir conectar a um SGBD MySql e consultar sua estrutura existente. As tabelas deste banco, suas colunas e relacionamentos deverão ser importados para o sistema.
RF012: O sistema deverá disponibilizar o cadastro de índices nas tabelas do banco de dados.
RF013: O sistema deverá disponibilizar a funcionalidade para elaboração dos diagramas da UML 2.0, casos de uso, classes e atividades.

Quadro 1 – Requisitos funcionais do sistema

O quadro 2 demonstra os requisitos não funcionais.

REQUISITOS NÃO FUNCIONAIS
RNF001: O sistema deverá ser compatível com a versão 6.0 ou superior do navegador Internet Explorer da Microsoft.
RNF002: O sistema deverá ser desenvolvido utilizando o SGBD MySql.
RNF003: O sistema deverá ser desenvolvido na linguagem Java para o ambiente web (JEE).
RNF004: O sistema deverá tratar dois tipos de usuários: administradores (controle total) e analistas de sistemas.
RNF005: O acesso ao sistema deverá ser permitido somente mediante informação de um código e uma senha de acesso.
RNF006: A senha de acesso deverá ser armazenada de forma criptografada no banco de dados.
RNF007: Os arquivos-texto gerados pelo sistema, com comandos SQL para criação e atualização do banco de dados, deverão ser gerados em conformidade com as particularidades de cada SGBD no que tange aos tipos de dados e outros comandos cuja sintaxe possa ser diferenciada entre eles.
RNF008: O sistema deverá, quando possível, utilizar AJAX para efetuar a troca de informações entre o cliente (navegador) e o servidor.
RNF009: O sistema deverá ser desenvolvido de acordo com a arquitetura MVC, separando claramente a interface das regras de negócio.

Quadro 2 – Requisitos não funcionais do sistema

3.2 ESPECIFICAÇÃO

Após a definição dos requisitos, foi desenvolvida a especificação do software, através da geração dos diagramas da UML com a ferramenta Enterprise Architect (EA) e do diagrama do modelo da base de dados com a ferramenta PowerDesigner.

Os diagramas desenvolvidos são apresentados nas seções a seguir.

3.2.1 Diagrama de casos de uso

De acordo com o requisito não funcional RNF004, o sistema deve tratar dois tipos de usuário: administradores e analistas. Dessa forma, foram desenvolvidos dois diagramas de caso de uso: um com os casos de uso onde o único ator é o administrador e outro com os

casos de uso comuns aos dois atores.

Os casos de uso exclusivos do administrador aparecem na figura 9.

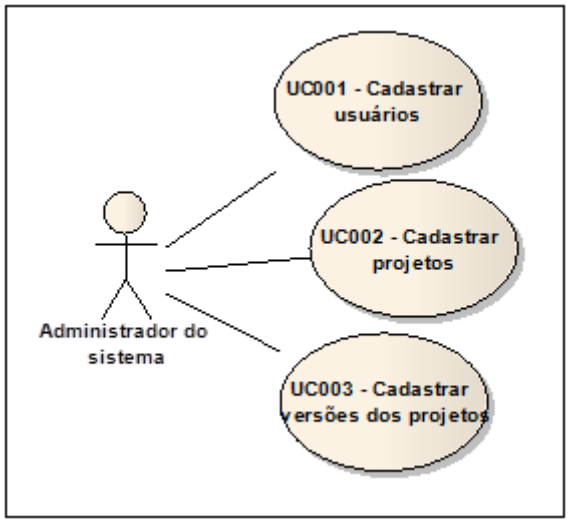


Figura 9 - Diagrama dos casos de uso exclusivos do administrador

Nos casos de uso desenvolvidos para o administrator do sistema, são devinidas as atividades relacionadas ao administrador. O UC001 define que o administrador pode cadastrar usuários no sistema, o UC002 define que o administrador pode cadastrar projetos ao sistema e o UC003, devine que o cadastro de versões. Com isto o sistema permite uma ligação entre usuários, projetos e suas versões.

A figura 10 ilustra os casos de uso comuns ao administrador e ao analista de sistemas.

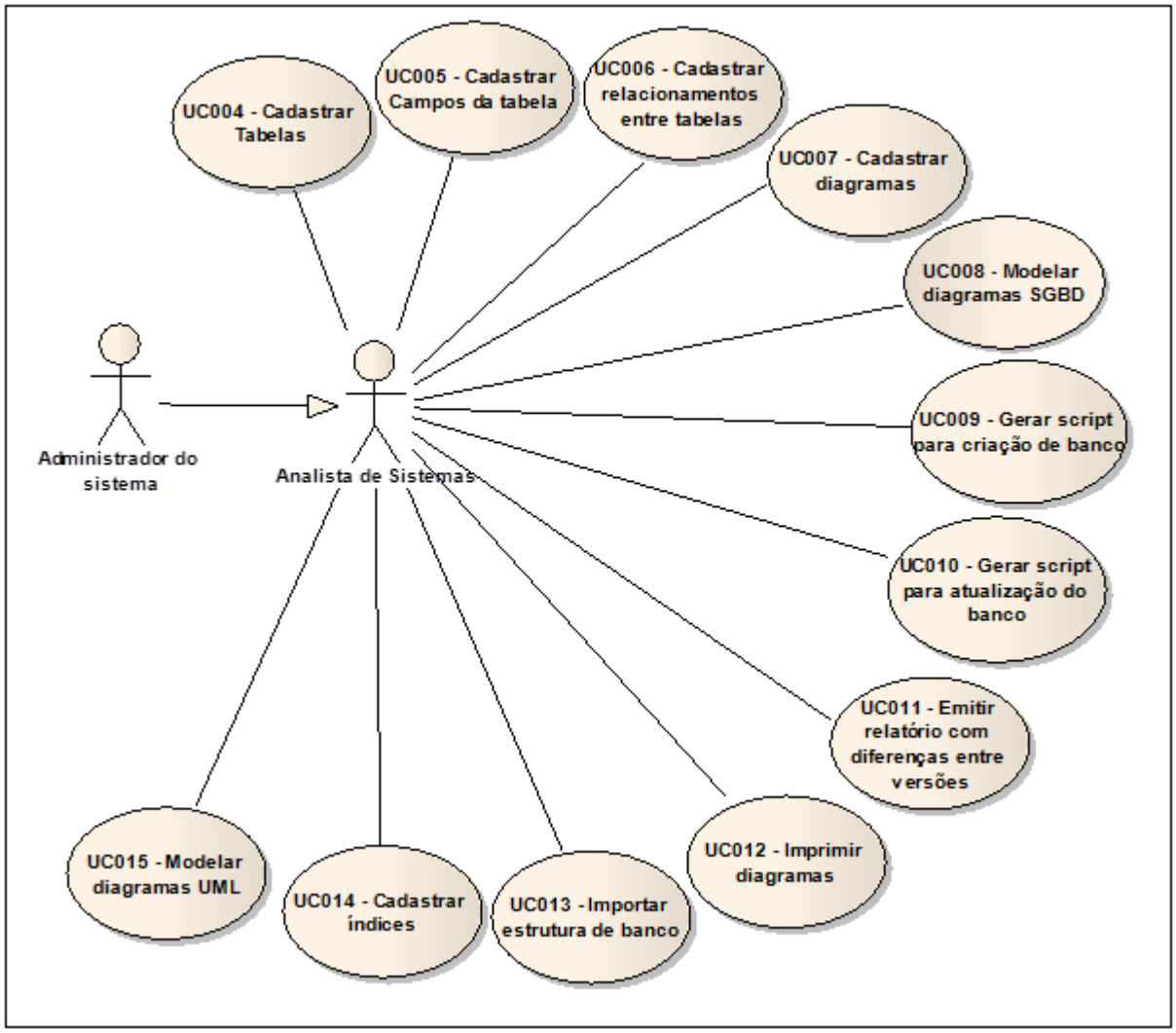


Figura 10 - Diagrama de casos de uso comuns ao administrador e ao analista

Os cenários dos casos de uso, desenvolvidos para o aplicativo, podem ser vistos no ANEXO D deste trabalho.

O quadro 3, a seguir, ilustra a matriz de relacionamento entre os requisitos funcionais e os casos de uso, cujo objetivo é garantir que todos os requisitos estejam sendo tratados por algum caso de uso, possibilitando também o rastreamento da relação entre eles.

	UC 001	UC 002	UC 003	UC 004	UC 005	UC 006	UC 007	UC 008	UC 009	UC 010	UC 011	UC 012	UC 013	UC 014	UC 015
RF001	■														
RF002		■													
RF003	■	■													
RF004			■												
RF005				■	■	■	■	■							
RF006									■						
RF007										■					
RF008											■				
RF009												■			
RF010								■							
RF011													■		
RF012														■	
RF013															■

Quadro 3 – Matriz de relacionamento entre os casos de uso e os requisitos funcionais

3.2.2 Diagramas de atividades

Nesta seção são listados os diagramas de atividades que demonstram visualmente o funcionamento dos casos de uso do sistema, no que tange à interação do usuário com o aplicativo.

Para a elaboração dos diagramas de atividades foi utilizada a ferramenta Enterprise Architect. A Figura 11 mostra o diagrama de atividades, relacionado ao caso de uso UC001 – Cadastrar usuários.

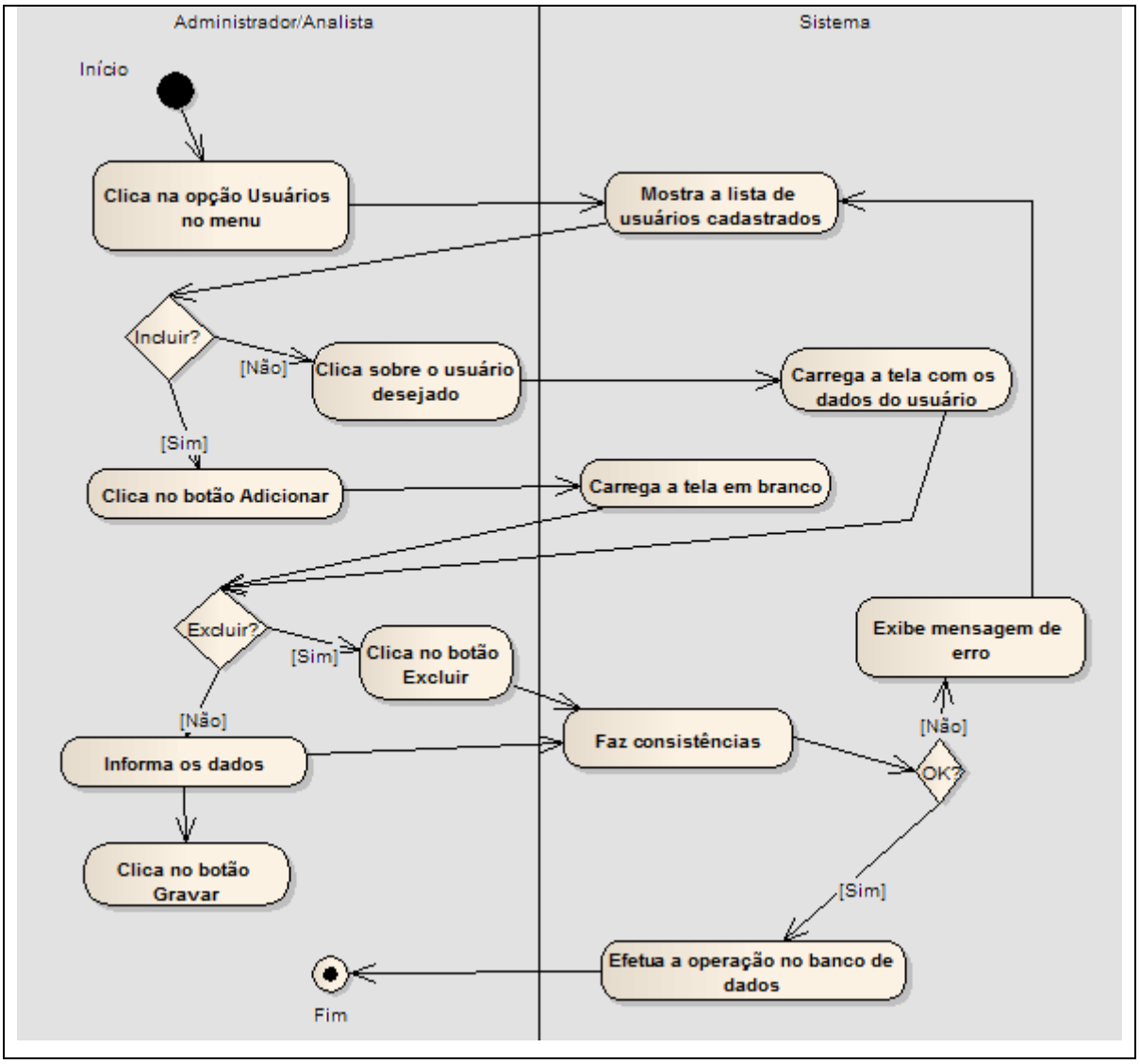


Figura 11 – Diagrama de atividades do caso de uso UC001 (cadastrar usuário)

Os demais diagramas de cadastro do sistema são muito parecidos ao diagrama do caso de uso UC001, por este motivo não serão mostrados neste trabalho.

A figura 12 mostra o diagrama de atividades do caso de uso UC009 – Gerar scripts para criação de banco.

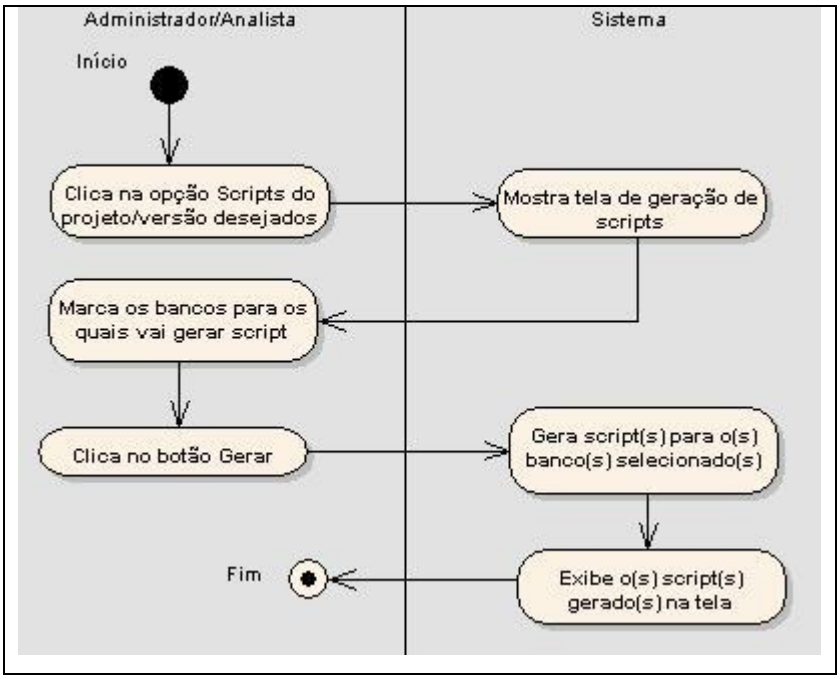


Figura 12 – Diagrama de atividades do UC009 (gerar script de criação)

A figura 13 mostra o diagrama de atividades do módulo de engenharia reversa, que está especificado pelo caso de uso UC013 - importa estrutura de banco.

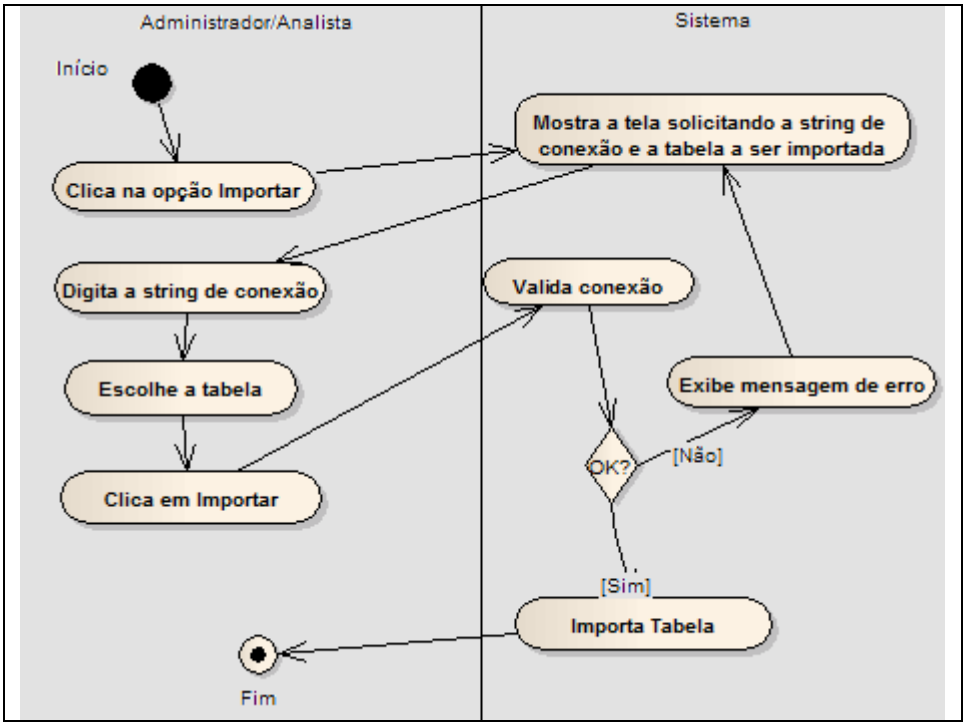


Figura 13 – Diagrama de atividades do UC013 (importa estrutura de banco)

A figura 14 mostra o diagrama de atividades do módulo de diagramas UML, que está especificado pelo caso de uso UC015 - modelar diagramas UML.

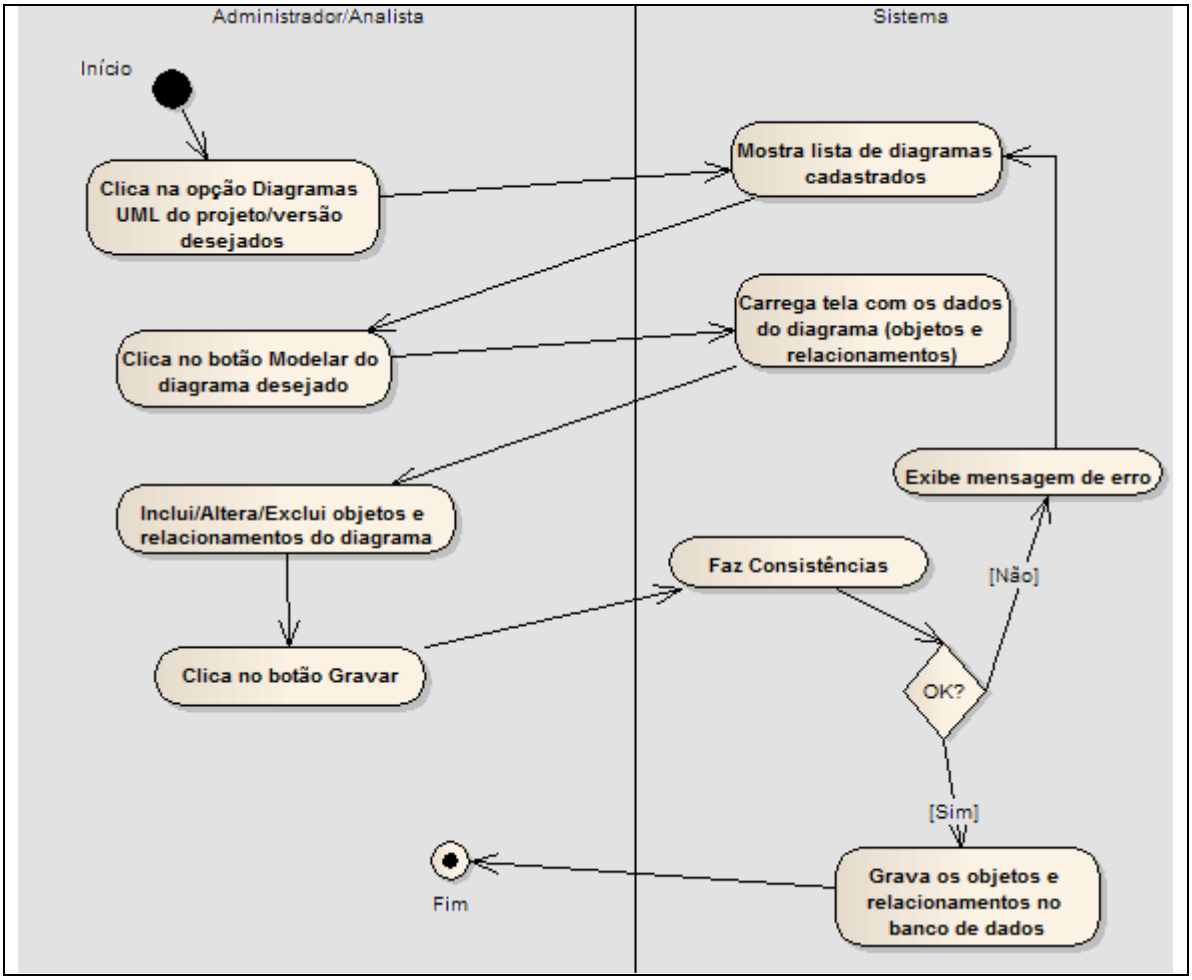


Figura 14 – Diagrama de atividades do UC015 (modelar diagramas UML)

3.2.3 Diagrama Entidade Relacionamento

Na figura 15, pode ser visualizado o diagrama entidade relaciomento do do sistema desenvolvido. Para geração deste diagrama foi utilizada a ferramenta Mysql Workbench.

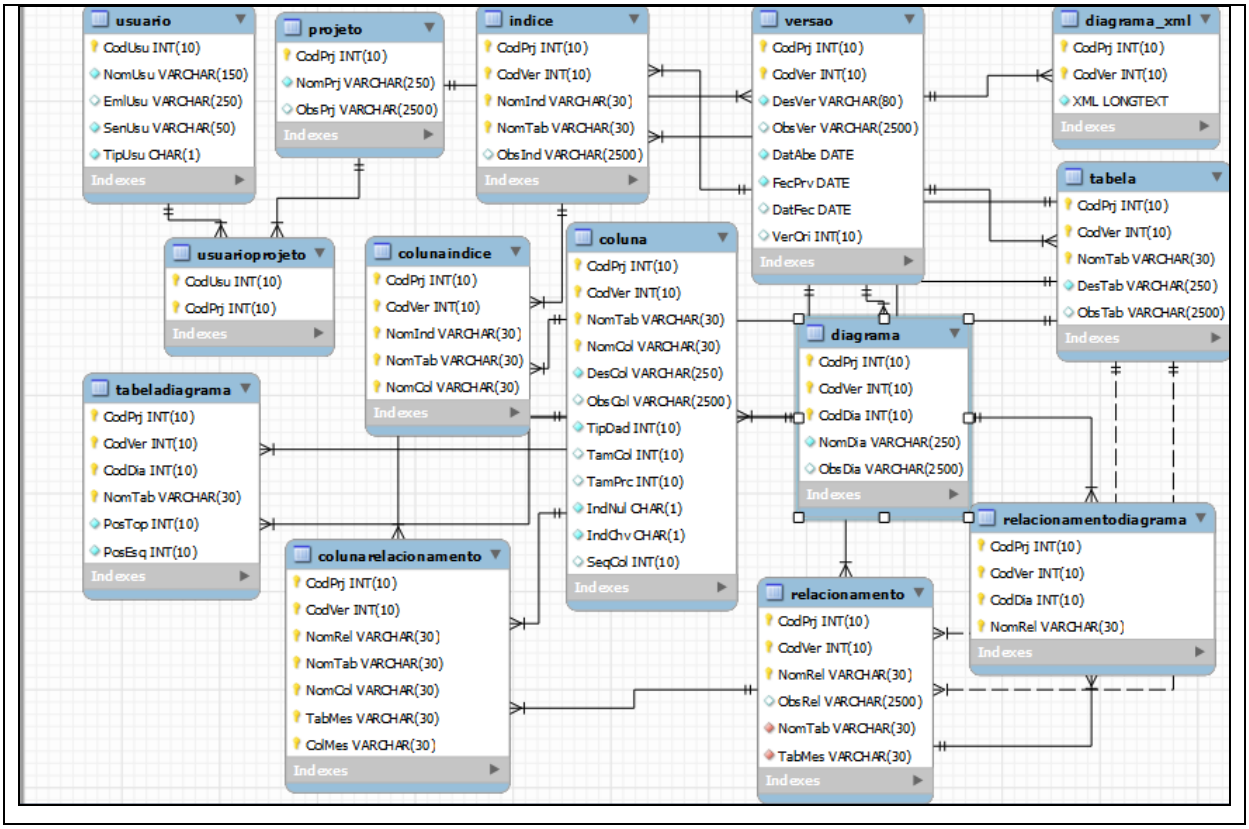


Figura 15 – Diagrama Entidade Relacional

3.2.4 Diagramas de pacotes e de classes

O sistema foi desenvolvido utilizando o conceito de orientação a objetos e, portanto, foram desenvolvidas várias classes, as quais foram agrupadas em pacotes para facilitar a organização e manutenção do código-fonte.

A figura 16 ilustra os principais pacotes do sistema, organizados de forma a demonstrar sua posição dentro da arquitetura MVC, que é um modelo de desenvolvimento de software. O modelo isola a lógica da aplicação, da interface do usuário, permitindo desenvolver, editar e testar separadamente cada parte.

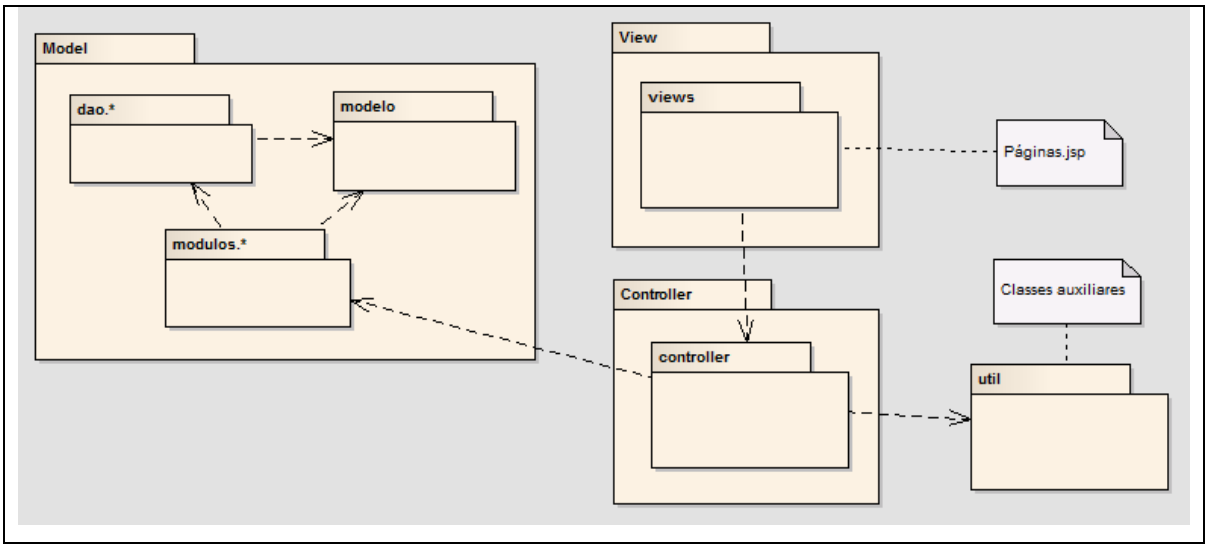


Figura 16 – Diagrama de pacotes do sistema na arquitetura MVC

A figura 17 detalha o pacote `dao.*` ilustrado na figura anterior.

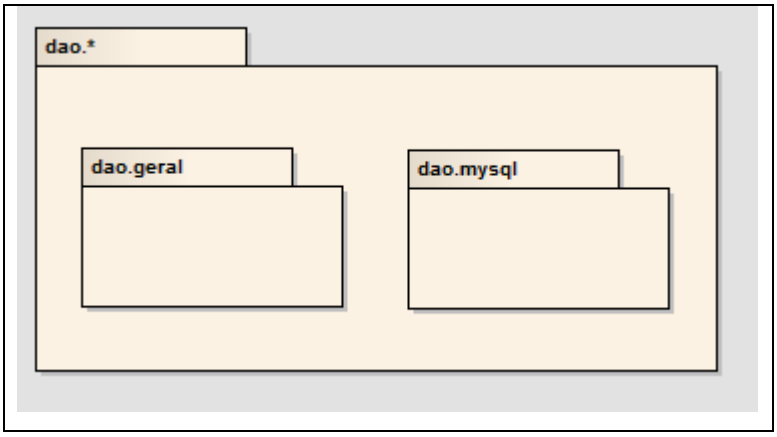


Figura 17 – Diagrama de pacotes detalhando o pacote `dao.*`

Por fim, a figura 18 detalha o pacote `modulos.*`.

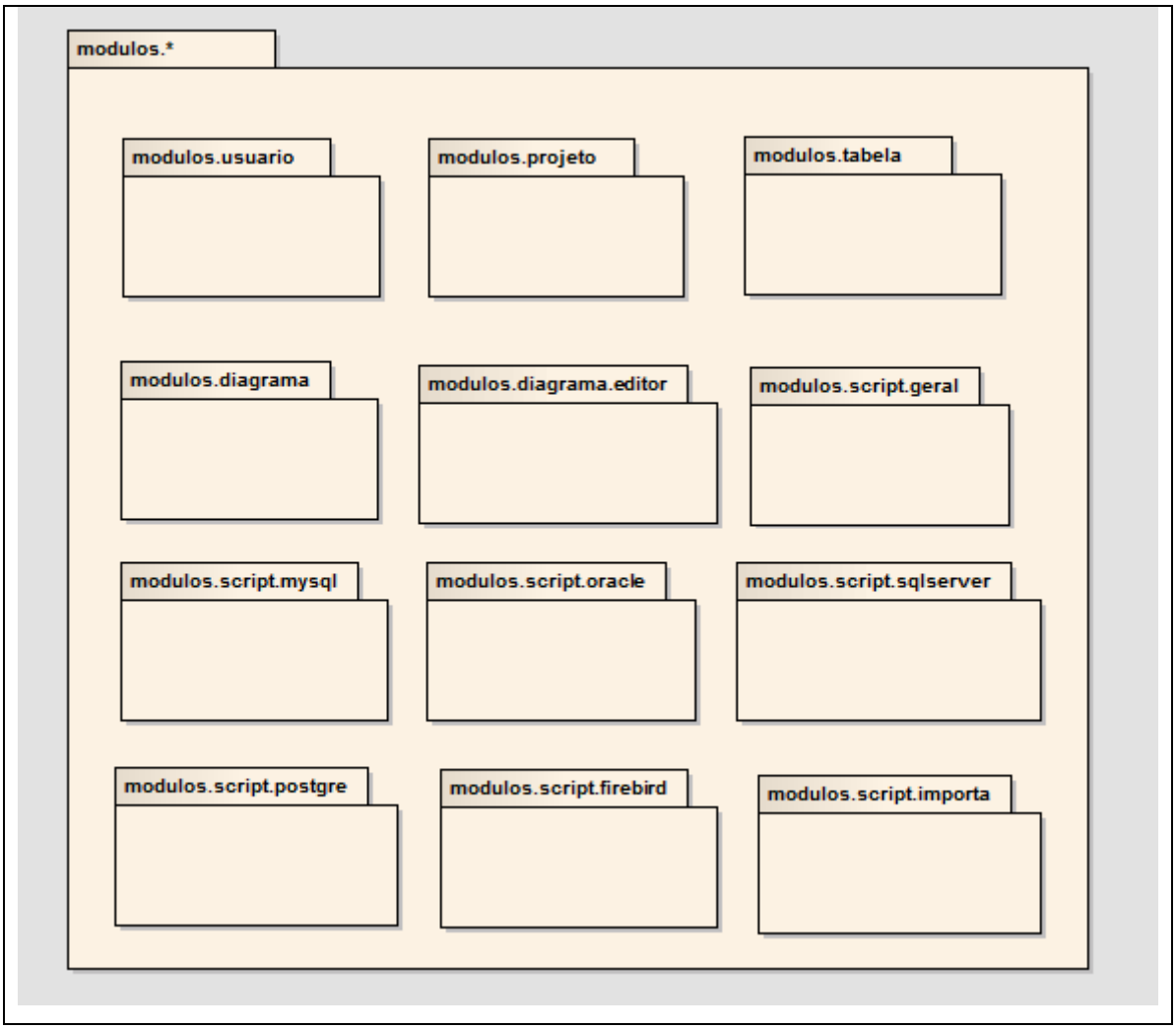


Figura 18 – Diagrama de pacotes detalhando o pacote `modulos.*`

O quadro 4 descreve a função de cada pacote dentro do sistema.

Pacote	Descrição
Views	Pacote onde estão as páginas.jsp, ou seja, a interface do sistema.
Controller	Pacote com as classes onde está implementada a camada de Controle da aplicação, ou seja, as classes que ligam o Modelo à Visão.
Útil	Classes diversas utilizadas para tratamento de exceções, criptografia, etc.
modelo	Neste pacote estão as classes do domínio da aplicação.
dao.geral	Classes abstratas para implementação do padrão DAO, responsáveis pelo acesso ao banco de dados.
dao.mysql	Contêm as subclasses DAO para acesso ao banco de dados MySql.
modulos.usuario	Possui as classes responsáveis pelas rotinas relacionadas ao cadastramento de usuários do sistema.
modulos.projeto	Neste pacote estão as classes responsáveis pelas rotinas relacionadas ao cadastramento e controle dos projetos.
modulos.tabela	Classes responsáveis pelas rotinas relacionadas ao cadastro de tabelas, colunas e relacionamentos.
modulos.diagrama	Pacote onde estão as classes relacionadas ao cadastro e modelagem dos diagramas.
modulos.diagrama.editor	Pacote onde estão as classes relacionadas ao cadastro e modelagem de diagramas UML.
modulos.script.geral	Neste pacote estão as classes genéricas utilizadas nas rotinas de geração dos *scripts* de criação e atualização de bancos de dados. Além disso, possui a classe que gera o relatório de diferenças entre as versões.
modulos.script.oracle	Classes específicas para geração dos *scripts* de atualização e criação do banco de dados para Oracle.
modulos.script.sqlserver	Classes específicas para geração dos *scripts* de atualização e criação do banco de dados para MSSqlServer.
modulos.script.mysql	Classes específicas para geração dos *scripts* de atualização e criação do banco de dados para Mysql.
modulos.script.postgreSql	Classes específicas para geração dos *scripts* de atualização e criação do banco de dados para PostgreSql.
modulos.script.firebird	Classes específicas para geração dos *scripts* de atualização e criação do banco de dados para Firebird.
modulos.script.importa	Classes responsáveis pela importação de modelo, tabelas, relacionamentos e índices de um banco de dados Mysql.

Quadro 4 – Funções dos pacotes no sistema

O quadro seguinte, 5, traz as classes do pacote `controller`.

Classe	Descrição
`FrontController`	Classe que implementa o padrão de mesmo nome, representando a camada de Controle da arquitetura MVC. Analisa as solicitações recebidas (endereços enviadas pelo *browser*), instancia e executa os `Commands` responsáveis por atender as solicitações.

Quadro 5 – Classes do pacote `controller`

No pacote `util` encontram-se as classes descritas no quadro 6.

Classe	Descrição
`Command`	Classe básica para implementação do padrão de projeto de mesmo nome. É estendida nos pacotes dos módulos.
`Criptografia`	Classe utilizada para criptografar a senha dos usuários do sistema.
`RequestHelper`	Classe abstrata com a interface padrão para controle dos atributos que deverão ser passados como parâmetros para a execução dos `Commands`.
`HttpRequestHelper`	Implementação da classe anterior, voltada para aplicações web.
`ViewFlow`	Classe para armazenamento das relações entre as solicitações, os `Commands` que devem ser chamados para atendê-las e as páginas .jsp que devem ser exibidas após a execução do respectivo `Command`.
`WmException`	Classe de exceção estendida de `Exception`.

Quadro 6 – Classes do pacote `util`

As classes do pacote `modelo` são descritas no quadro 7.

Classe	Descrição
`ChavePrimaria`	Classe do domínio da aplicação, com os atributos referentes às chaves primárias das tabelas.
`Coluna`	Classe do domínio da aplicação, com os atributos referentes às colunas das tabelas.
`ColunaRelacionamento`	Classe do domínio da aplicação, com os atributos referentes às colunas pertencentes a um relacionamento.
`Diagrama`	Classe do domínio da aplicação, com os atributos referentes aos diagramas.
`Projeto`	Classe do domínio da aplicação, com os atributos referentes aos projetos.
`Relacionamento`	Classe do domínio da aplicação, com os atributos referentes aos relacionamentos entre tabelas.
`Tabela`	Classe do domínio da aplicação, com os atributos referentes às tabelas.
`TabelaDiagrama`	Classe do domínio da aplicação, com os atributos referentes às tabelas que integram os diagramas.
`Usuario`	Classe do domínio da aplicação, com os atributos referentes aos usuários do sistema.
`UsuarioProjeto`	Classe do domínio da aplicação, com os atributos referentes às relações entre usuários e projetos.
`Versao`	Classe do domínio da aplicação, com os atributos referentes às versões dos projetos.

Quadro 7 – Classes do pacote `modelo`

O quadro 8 descreve as classes dos pacotes `dao.geral`, que são estendidas no pacote `dao.mysql`.

Classe	Descrição
ColunaDAO	Classe com métodos para gravação, exclusão e recuperação de dados referentes às colunas das tabelas.
ColunaRelacionamentoDAO	Classe com métodos para gravação, exclusão e recuperação de dados referentes às colunas pertencentes aos relacionamentos entre as tabelas.
ComparaVersoesDAO	Classe com métodos para recuperação de dados referentes às diferenças existentes entre duas versões de um projeto (tabelas, colunas e relacionamento incluídos, excluídos e/ou alterados).
DBDAOFactory	Classe com métodos de criação dos objetos DAO que serão utilizados pelo sistema. Classe-fábrica.
DiagramaDAO	Classe com métodos para gravação, exclusão e recuperação de dados referentes aos diagramas.
ProjetoDAO	Classe com métodos para gravação, exclusão e recuperação de dados referentes aos projetos.
RelacionamentoDAO	Classe com métodos para gravação, exclusão e recuperação de dados referentes aos relacionamentos entre as tabelas.
TabelaDAO	Classe com métodos para gravação, exclusão e recuperação de dados referentes às tabelas.
UsuarioDAO	Classe com métodos para gravação, exclusão e recuperação de dados referentes aos usuários.
UsuarioProjetoDAO	Classe com métodos para gravação, exclusão e recuperação de dados referentes aos relacionamentos entre usuários e projetos.
VersãoDAO	Classe com métodos para gravação, exclusão e recuperação de dados referentes às versões dos projetos.

Quadro 8 – Classes do pacote `dao.geral`, estendidas no pacote `dao.mysql`

Nessa etapa do desenvolvimento do trabalho, também foi criado o diagrama de classes relativo ao núcleo de processamento das requisições no sistema, representado na figura 19.

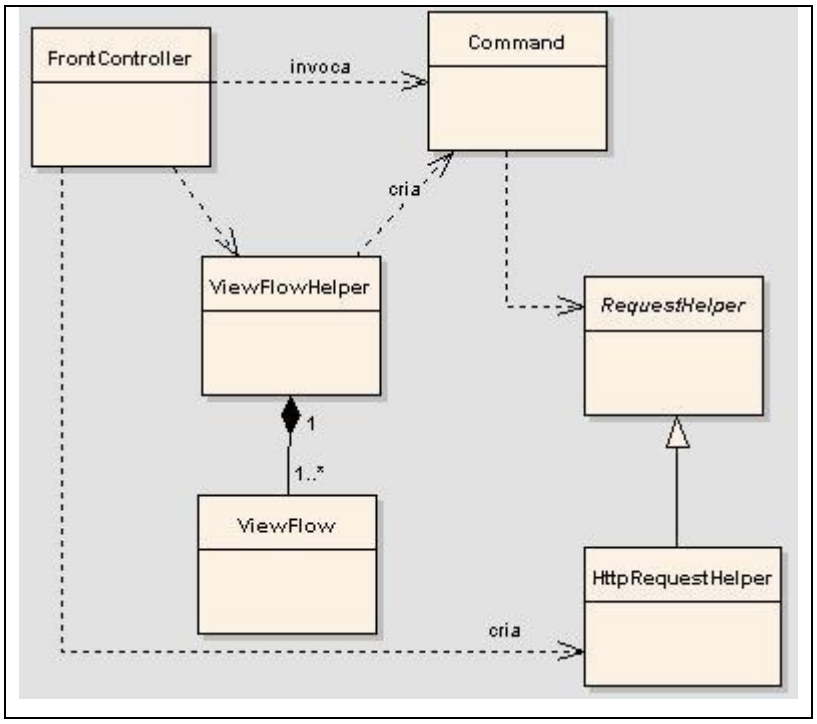


Figura 19 – Classes do núcleo de processamento de requisições do sistema

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas técnicas e ferramentas utilizadas no desenvolvimento do software especificado no tópico anterior (batizado como WebModeler 2.0).

3.3.1 Técnicas e ferramentas utilizadas

O software foi desenvolvido na linguagem Java, de acordo com a especificação JEE, sendo que o ambiente de desenvolvimento utilizado foi o Eclipse, na versão 4.3.0. O Eclipse foi utilizado para desenvolver todas as camadas (MVC) do aplicativo, incluindo a interface (arquivos JSP) e seus códigos JavaScript. O servidor de aplicações utilizado para executar o software foi o Apache Tomcat na versão 7.0 e o SGBD utilizado foi o MySql na versão 5.1.11.

3.3.2 Implementação do protótipo

A implementação do protótipo ocorreu conforme os seguintes passos:

a) desenvolvimento da camada de persistência e recuperação dos dados;
b) desenvolvimento do núcleo de processamento de requisições do sistema;
c) implementação dos cadastros básicos do sistema;
d) implementação do módulo de modelagem gráfica;
e) implementação do módulo de geração de *scripts*;
f) definição dos estilos (cores, fontes) da interface da aplicação;
g) implementação de geração de *scripts* para demais SGBD relacionais;
h) implementação do módulo de impressão de diagramas;
i) implementação do cadastro de índices;
j) implementação do módulo de importação de modelo de SGBD relacional;
k) implementação do módulo de modelagem de diagramas UML.

Os passos de letra “a)” ao “f)” foram desenvolvidos por Bachmann (2007). Os passos “g)” ao “k)” serão detalhados a seguir.

3.3.2.1 Implementação de geração de *scripts* para demais SGBD relacionais

O desenvolvimento deste módulo foi dividido nas seguintes etapas:

a) implementação das classes e rotinas responsáveis pela geração dos *scripts* de criação do banco de dados para os SGBD Oracle, MSSqlServer, Mysql, PostgreSql e Firebird;
b) implementação das classes e rotinas para geração dos *scripts* de atualização do banco de dados para os mesmos SGBD supra citados;
c) criação das classes e rotinas para geração do relatório de diferenças entre versões.

As rotinas foram desenvolvidas de forma a gerar *scripts* compatíveis com a versão 10g do Oracle, com a versão 2000 do MSSqlServer, Mysql com versão 5, PostgreSql com versão 9 e Firebird com versão 2.5.1, sendo que os testes durante a implementação foram feitos nestas versões destes SGBD.

A tela do sistema onde são cadastradas tabelas e suas colunas foi desenvolvida de

forma que o usuário informe tipos de dados próprios do sistema, com nomenclaturas em Língua Portuguesa (Numérico, Texto Variável, Data, etc.), ao invés de tipos próprios de algum SGBD. Sendo assim, o primeiro passo foi relacionar os tipos do sistema com os tipos nativos do Oracle, do MSSqlServer, do Mysql, do PostgreSql e do Firebird, relacionamento este feito com base nas informações de Oracle (2005), Battisti (2001, p.155-157), Mysql (2012), PostgreSql (2012) e Firebird (2012) e descrito no quadro 9.

WebModeler 2.0	**Oracle**	**MSSqlServer**	**Mysql**	**PostgreSql**	**Firebird**
Texto Variável	Varchar2	Varchar	Varchar	Varchar	Varchar
Texto Fixo	Char	Char	Char	Char	Char
Inteiro	Integer	Int	Int	Int	Integer
Inteiro Curto	Smallint	Smallint	Smallint	Smallint	Smallint
Inteiro Longo	Integer	Bigint	Bigint	Bigint	Bigint
Numérico	Decimal	Decimal	Decimal	Numeric	Numeric
Data	Date	DateTime	Date	Date	TimesTamp
Data/Hora	DateTime	DateTime	DateTime	Time	TimesTamp
Binário	Blob	Binary	Blob	Bytea	Char

Quadro 9 – Relacionamento entre os tipos de dados do sistema e dos SGBD

Uma vez definido o relacionamento entre os tipos de dados, foram desenvolvidas as classes e rotinas para geração dos scripts de criação do banco de dados. Foram criadas as classes, `ScriptsOracle`, `ScriptsSqlServer`, `ScriptMysql`, `ScriptPostgreSql` e `ScriptFirebird`, as quais possuem métodos que retornam o texto adequado para criação, alteração e exclusão das tabelas, relacionamentos e colunas. Ambas possuem implementação bastante parecida e retornam comandos SQL idênticos, porém, decidiu-se criar classes separadas prevendo uma futura melhoria no sistema, onde poderiam ser disponibilizados recursos mais avançados e específicos de cada banco de dados.

O quadro 10 exibe o trecho do código-fonte da classe `ScriptsMysql` onde é gerado o script de criação de uma tabela para o banco de dados Mysql.

```
public String scriptGeraTabela(Tabela tab) throws WmException{
	String retorno = "";
	String create = "create table ";
	create +=  tab.getNome();
	create += " (";
	retorno += create;

	//carregar as colunas
	ColunaDAO colDAO = this.factory.getColunaDAO();
	List<Coluna>   listaColunas   =   colDAO.listar(tab.getProjeto(),   tab.getVersao(),
tab.getNome());
	Coluna col;
	String coluna;
	String pk = "";
	for (int i = 0; i < listaColunas.size(); i ++){
		col = listaColunas.get(i);
		if	(col.isPertenceChave()){
			pk += col.getNome();
			pk += ",";
		}
		coluna = "  \n  ";
		coluna += col.getNome();
		coluna += " ";

		coluna	+=	this.tipoDado(col.getTipoDado(),	col.getTamanho(),
col.getPrecisao());
		if	(!col.isPermiteNulo())
			coluna += " not null,";
		else
			coluna += ",";
		retorno += coluna;
		//retorno += "\n";
	}

	if	(!pk.equals("")){
		pk = pk.substring(0, pk.length() - 1); //tira a última vírgula
		coluna = "\n constraint CP_";
		coluna += tab.getNome();
		coluna += " primary key (";
		coluna += pk;
		coluna += ")";
		retorno += coluna;
	} else
		retorno = retorno.substring(0, retorno.length() - 1); //tira a última vírgula
	retorno += ") ;";

	return retorno;
}
```

Quadro 10 – Código-fonte da geração de *script* de criação de tabela para o Mysql

A lógica implementada para geração destes *scripts* é dividida em duas partes: primeiramente são inseridos os comandos para criar todas as tabelas e logo após os comandos para criar todos os relacionamentos. No quadro anterior pode ser visto que neste caso foram utilizadas as classes DAO para ter acesso às informações do banco de dados. Inicialmente são buscadas as colunas da tabela e para cada tabela monta uma linha na variável de retorno, contendo o nome da coluna, seu tipo e se ela tiver sido cadastrada indicando que não poderá ter valores nulos, concatenar-se-á o texto a expressão *not null*. A cada coluna da lista a classe também verifica se a mesma pertence à chave primária da tabela e, se pertencer, adiciona o seu nome na variável que será utilizada para montar o texto relativo à criação desta chave.

A sintaxe utilizada pelo sistema para gerar os comandos de criação das tabelas e dos

relacionamentos pode ser vista nos quadros 11 e 12.

```
create table NOME_TABELA(
  NOME_COLUNA TIPO_DADO [not null],
  constraint CP_NOME_TABELA primary key (COLUNAS_PK)
)
```

Quadro 11 – Sintaxe para criação das tabelas

```
alter table NOME_TABELA
  add constraint NOME_RELACIONAMENTO foreign key (COLUNAS)
     references TABELA_MESTRE (COLUNAS_MESTRE)
```

Quadro 12 – Sintaxe para criação dos relacionamentos

O *script* de atualização gera um arquivo PDF com os comandos SQL necessários para atualizar a estrutura o SGBD que esteja de acordo com uma versão mais antiga do projeto, igualando esta estrutura à da versão mais recente. A implementação deste item segue a lógica:

a) inicialmente são inseridos os comandos necessários para excluir relacionamentos entre tabelas que existiam na versão antiga e que foram removidos na versão atual;
b) logo após, os comandos para excluir relacionamentos que foram alterados de uma versão para outra;
c) na sequência, comandos para excluir chaves primárias que foram removidas ou alteradas;
d) o passo seguinte é inserir os comandos de criação das tabelas que foram incluídas na versão atual;
e) a seguir, os comandos para excluir tabelas removidas na versão atual do projeto;
f) depois vêm os comandos para incluir as novas colunas, excluir as colunas removidas e modificar as colunas alteradas;
g) os comandos para recriar as chaves primárias incluídas ou alteradas são gerados a seguir;
h) por fim, os comandos para recriar os relacionamentos incluídos ou alterados.

Para gerar estes scripts, foram desenvolvidas outras classes: `ScriptAtualizacaoOracle`, `ScriptAtualizacaoSqlServer`, `ScriptAtualizacaoMySql`, `ScriptAtualizacaoPostgreSql` e `ScriptAtualizacaoFirebird`, que utilizam as classes `ScriptsOracle`, `ScriptsSqlServer`, `ScriptsSqlMySql`, `ScriptsSqlPostgreSql` e `ScriptsSqlFirebirde`, assim como estas últimas, também possuem implementação semelhante mas foram criadas separadas para facilitar a futura implementação de recursos próprios de cada banco. O quadro 13 mostra parte do código-fonte da classe

ScriptAtualizacaMySql.

```
Package modulos.script.mysql;
[...]
public class ScriptAtualizacaoMySql {
[...]
  public ScriptAtualizacaoMySql(DBDAOFactory factory) {
        super();
        this.factory = factory;
        this.compara = this.factory.getComparaVersoesDAO();
        this.geraScript = new ScriptsMySql(this.factory);
 }
  public String gerar(int projeto,int vrsOrigem, int vrsDestino) throws WmException{
    String retorno = "";
    List<Relacionamento>relExc = this.compara.getRelExcl(projeto,vrsOrigem,vrsDestino);
    [...]
    List<Tabela> tabNov = this.compara.getTabelasNovas(projeto,vrsOrigem, vrsDestino);
    [...]
    //Primeiro, excluir relacionamentos removidos na versao destino
    for (i = 0; i < relExc.size(); i ++){
        retorno += geraScript.scriptExcluiRelacionamento(relExc.get(i));
        retorno += "\n\n";
    }
    [...]
    //Quarto, excluir tabelas removidas
    for (i = 0; i < tabExc.size(); i ++){
        retorno += geraScript.scriptExcluiTabela(tabExc.get(i));
        retorno += "\n\n";
    }
    [...]
    //Nono, criar relacionamentos alterados
    for (i = 0; i < relAlt.size(); i ++){
        retorno += geraScript.scriptGeraRelacionamento(relAlt.get(i));
        retorno += "\n\n";
    }
    return retorno;
  }
}
```

Quadro 13 – Código-fonte da classe `ScriptAtualizacaoMySql`

Pode-se observar no quadro anterior a utilização da classe `ComparaVersoesDAO`, que foi implementada para acessar o banco de dados do sistema, comparar as versões antigas com a mais atual para a geração do *script* e carregar a listas contendo as tabelas, relacionamentos e colunas que foram incluídas, modificadas ou excluídas na versão atual.

3.3.2.2 Implementação do módulo de impressão de diagramas

Este módulo do sistema, foi dividido em duas etapas de acordo com a funcionalidade:

a) alterar a tela de diagrama do banco de dados;
b) permitir a impressão do diagrama de modelagem.

Para o desenvolvimento da funcionalidade de alteração de diagramas foi incluída as funções `zoomin` e `zoomout` na página dia_desenho.jsp. A funcionalidade de impressão foi implementada nesta mesma página com a função `myprint`. O quadro 14 demonstra parte do código-fonte destes recursos.

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=windows-1252">
<link rel="stylesheet" href="css/wm.css" type="text/css"/>
<title>Diagramas</title>
<%@ page contentType="text/html; charset=ISO-8859-1" %>
<script type="text/javascript">
<%@include file="../javascripts/dragdrop.js" %>
<%@include file="../javascripts/ajax.js" %>
[...]
function zoomin(){
        var y=document.body.clientHeight;
        var x=document.body.clientWidth;
        factor=factor+0.1;
        return document.body.style.zoom=factor;
        }
function zoomout(){
        var y=document.body.clientHeight;
        var x=document.body.clientWidth;
        factor=factor-0.1;
        return document.body.style.zoom=factor;
        }
function myprint() {
        window.parent.princp.focus();
        window.print();
        }
[...]
</script>
[...]
<body onclick="escondeDivMenuTabela();" oncontextmenu="return false" class="body_desenho">
[...]
<button id="btZoomIn" style="width: 30px; height: 30px; background-color: #FFFFFF"
onclick="zoomin();">
   <img border="0" src="img/Zoom_In.png" width="20" height="20" alt="Zoom +" style="cursor:
default">
</button>
<button id="btZoomOut" style="width: 30px; height: 30px; background-color: #FFFFFF"
onclick="zoomout();">
   <img border="0" src="img/Zoom_Out.png" width="20" height="20" alt="Zoom -" style="cursor:
default">
</button>
<button id="btPrint" style="width: 30px; height: 30px; background-color: #FFFFFF"
onclick="myprint();">
   <img border="0" src="img/print_icon.png" width="20" height="20" alt="Imprimir"
style="cursor: default">
</button>
[...]
</script>
</body>
</html>
```

Quadro 14 – Código fonte da página `dia_desenho.jsp`

Conforme exibido no quadro anterior, não há código Java na página, sendo que o mesmo foi substituído por código *Javascript*, que é executado diretamente no cliente, ou seja, o navegador que executa seu código.

3.3.2.3 Implementação do cadastro de índices

Este módulo do sistema, visa o cadastro de índices para as tabelas do projeto. Indices são muito utilizados para melhorar o desempenho em consultas a banco de dados. O quadro 15 mostra o trecho do código-fonte deste recurso.

```
package modulos.tabela;
[...]
public class GravarIndiceCommand extends Command {
	@Override
	public String executar(RequestHelper rh) throws WmException {
		int projeto = Integer.parseInt(rh.getParametro("projeto"));
		int versao = Integer.parseInt(rh.getParametro("versao"));
		String nome = rh.getParametro("nome");
		String old = rh.getParametro("oldNome");
		String observacao = "";
		String insChave = rh.getParametro("insereChave");
		try {
			observacao = URLDecoder.decode(rh.getParametro("observacao"), "UTF-8");
		} catch (UnsupportedEncodingException e) {
			throw new WmException(e.getMessage());
		}
		[...]
		String tabela = rh.getParametro("tabela");
		String[] colunas = rh.getParametroLista("colunas");
		boolean inclusao = (rh.getParametro("inclusao").equals("sim"));
		//gravar o indice
		Indice rel = new Indice();
		rel.setProjeto(projeto);
		rel.setVersao(versao);
		rel.setNome(nome);
		rel.setTabela(tabela);
		rel.setObservacao(observacao);
		[...]
		return "OK";
	}
}
```

Quadro 15 – Código-fonte da classe `GravarIndiceCommand`

Este módulo, insere o nome do índice e as colunas de uma tabela que necessitam ser indexadas.

3.3.2.4 Implementação do módulo de importação de modelo de SGBD relacional

Este módulo do sistema, permite a importação de tabelas de um banco de dados mysql, onde as tabelas são inseridas no sistema e é gerado um diagrama inicial com os relacionamentos e colunas das tabelas. A implementação deste módulo segue a lógica abaixo:

a) inicialmente se conecta ao banco com a *string* informada na tela. Nesta *string* deve conter usuário, senha e nome do banco de dados;
b) logo após, é realizada a consulta da tabela informada e seus relacionamentos;
c) em seguida, os dados consultados no banco de origem são inseridos no banco de dados do protótipo;
d) por fim, é criado o diagrama inicial com as tabelas importadas e seus relacionamentos.

Para o desenvolvimento deste módulo foi criada a classe `ImportaCommand` que estende a classe `Command`. O quadro 16 mostra parte do código-fonte da classe `ImportaCommand`.

```
package modulos.script.importa;
[...]
public class ImportaCommand extends Command {
[...]
  public String executar(RequestHelper rh) throws WmException {
  [...]
  try {
      con = DriverManager.getConnection(url, user, pass);
      st       = con.createStatement();
      st1 = con.createStatement();
      st2 = con.createStatement();
      rs = st.executeQuery("SHOW TABLES LIKE '"+ ImpTable + "'");
      if (rs.next()){
        ImpTable = rs.getString(1);
      }
      rs.close();
      rs  = st.executeQuery("DESCRIBE " + ImpTable);
      int i=0;
      while (rs.next()) {
          if(i==0){
          //---IMPORTA TABELA
          sql =
          "INSERT INTO TABELA (CodPrj,CodVer,NomTab,DesTab,ObsTab) " +
          " VALUES ( " + projeto
          + ", " + versao
          + ", '" + ImpTable + "'"
          + ", '" + ImpObs + "'"
          + ", '" + ImpObs + "')";
          try{
            st1.executeUpdate(sql);
            retorno += "[INFO]: Tabela ("+ ImpTable +") Importada<br>";
          }catch(Exception e1){
          retorno += "ERRO: Tabela (" + ImpTable + ") ja existem" + e1.getMessage();
          rh.setAtributo("texto",retorno);
          return "ERRO";
      }
    }
    [...]
    return "OK";
  }
  private Integer TrataCampo(String campo){
     if(campo.equalsIgnoreCase("VARCHAR"))   {
        return 1; } //texto variavel
     if(campo.equalsIgnoreCase("CHAR"))      {
        return 2; } //texto fixo
     if(campo.equalsIgnoreCase("INT"))       {
        return 3; } //inteiro
     if(campo.equalsIgnoreCase("TINYINT")  ||
        campo.equalsIgnoreCase("SMALLINT") ||
        campo.equalsIgnoreCase("MEDIUMINT")){
        return 4; } //inteiro curto
     if(campo.equalsIgnoreCase("BIGINT"))    {
        return 5; } //inteiro longo
     if(campo.equalsIgnoreCase("DECIMAL")    ||
        campo.equalsIgnoreCase("FLOAT")      ||
        campo.equalsIgnoreCase("DOUBLE")||
        campo.equalsIgnoreCase("REAL"))      {
        return 6; } //Numerico
     if(campo.equalsIgnoreCase("DATE"))      {
        return 7; } //Data
     if(campo.equalsIgnoreCase("DATETIME"))  {
        return 8; } //Data Hora
     if(campo.equalsIgnoreCase("BINARY"))    {
        return 9; } //Binario
     return 0;
     }
}
```

Quadro 16 – Código-fonte da classe `ImportaCommand`

Pode-se observar no quadro anterior a utilização de mapeamento dos tipos nativos do Mysql para os tipos do sistema, assim os dados inseridos seguem o padrão adotado pelo sistema.

Para a conexão em dois bancos Mysql na classe `ImportaCommand` foi criado o método executar que tem a declaração de duas strings de conexão e alterna as sessões entre os bancos de dados. O quadro 17 mostra o trecho de declaração de duas conexões do método `executar`.

```
package modulos.script.importa;
[...]
public class ImportaCommand extends Command {
[...]
        @Override
        public String executar(RequestHelper rh) throws WmException {
        [...]
             try {
             con = DriverManager.getConnection(url, user, pass);
             con2   =   DriverManager.getConnection("jdbc:mysql://localhost:3306/tcc",   user,
pass);
             st        = con.createStatement();
             st1 = con2.createStatement();
             [...]
                return "OK";
        }
        [...]
}
```

Quadro 17 – Trecho de código do método `executar`

3.3.2.5 Implementação do módulo de modelagem de diagramas UML

A implementação deste módulo, visa a disponibilização da modelagem de alguns diagramas UML. Para facilitar a implementação da modelagem gráfica foi utilizada a biblioteca `jsUML2`, ao qual, segundo Salguero (2012), permite desenvolver diagramas UML dinamicamente executando diretamente pelo *browser*. A biblioteca `jsUML2` de Salguero (2012), é desenvolvida em Javascript com html5 o que possibilita vários recursos gráficos ao browser. O quadro 18 demonstra a página inicial criada para o editor de modelagem gráfica de diagramas UML.

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="es" lang="es">
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
    <title>UML Diagram</title>
    <script src="js/modernizr-latest.js"></script>
    <link type="text/css" rel="stylesheet" href="css/UDStyle.css" media="screen" />
    <script type="text/javascript" src="js/UDCore.js"></script>
    <script type="text/javascript" src="js/UDModules.js"></script>
    <script type="text/javascript" src="js/UDApplication.js"></script>
    <script type="text/javascript">
       var app;
       var xmltmp;
      window.onload = function() {
                              var width = window.innerWidth - 250;
                              if(width < 400)  width = 400;
                              if(width > 1000) width = 1000;
        app = new Application( { id: 'umldiagram', width: 800, height: 500 } );
        LOAD();
      }
       function LOAD(){
       xml = '${xml}';
       if( xml.length > 15){
              app.setXMLString(xml);
       }
       }
       function SAVE(){
              var xml_input = document.getElementById('xml');
              var xml = (new DOMParser()).parseFromString( '<umldiagram/>', 'text/xml');
              xmltmp = app.getXMLString();
              xml_input.value = app.getXMLString();
              return true;
       }
    </script>
  </head>
  <body>
<form name='save' action="diag_save.do" method="POST"  onsubmit="return SAVE()">
       <input type="hidden" name="projeto" id="projeto" value="${projeto}"/>
       <input type="hidden" name="versao" id="versao" value="${versao}"/>
       <input type="hidden" name="xml" id="xml" value=""/>
       <input type="submit" name='EXP' value='Salvar'>
</form>
    <div id="umldiagram" >
    </div>
    <div id="debug"></div>
  </body>
</html>
```

Quadro 18 – Conteúdo do arquivo `diag_editor.jsp`

Os arquivos `UDCore.js`, `UDModules.js` e `UDApplication.js` implementam as funcionalidades disponibilizadas pela biblioteca `jsUML2` de Salgueiro (2012). A função Save grava os diagramas no banco de dados e sempre quando houver um diagrama salvo para a versão do projeto, quando se clica na opção Editor de Diagramas no menu do sistema, estes diagramas salvos são carregados.

Para salvar os diagramas no banco de dados o conteúdo do editor de diagramas é salvo em um arquivo XML, que é gravado no banco de dados. Neste XML são guardados os diagramas, os componentes destes diagramas, junto com seus relacionamentos e posições. O quadro 19 demonstra a classe `SaveCommand`, criada para salvar os diagramas no banco de dados.

```
package modulos.diagrama.editor;
[...]
public class SaveCommand extends Command {
[...]
	@Override
	public String executar(RequestHelper rh) throws WmException {
		int projeto = Integer.parseInt(rh.getParametro("projeto"));
		int versao = Integer.parseInt(rh.getParametro("versao"));
		String xml = rh.getParametro("xml");
		DiagramaDAO diaDAO = this.getDbFactory().getDiagramaDAO();
		Diagrama dia = new Diagrama();
		dia.setProjeto(projeto);
		dia.setVersao(versao);
		dia.setXml(xml);
		diaDAO.gravarxml(dia);//grava a versão do projeto no banco
		rh.setAtributo("texto","Diagramas Salvos");
		return "OK";
	}
}
```

Quadro 19 – Código fonte da classe `SaveCommand`

3.3.2.6 Operacionalidade da implementação

Aqui será demonstrado o funcionamento da implementação através de um estudo de caso de uma situação hipotética, no qual uma empresa de reciclagem de materiais contratou um analista de sistemas para especificar uma melhoria de seu software Gestão de Reciclados (GR), porém esta empresa de reciclagem não possui documentação alguma sobre este sistema. O analista irá usar o sistema desenvolvido neste trabalho para realizar uma engenharia reversa no banco de dados e obter um modelo básico do mesmo.

O primeiro passo é o cadastramento do projeto e do analista, por parte do administrador do sistema, o qual acessará o software informando seu código e senha na tela de entrada exibida na figura 20.

Figura 20 - Tela de entrada do sistema

Se o código e a senha informados estiverem corretos, será apresentada a tela principal do sistema, conforme a figura 21.

Figura 21 - Tela principal do sistema para o administrador

Através do item de menu Administrador, serão acessadas as telas de cadastro de projetos e usuários. Já no item Analista, estão todos os modelos de todos os bancos de dados projetados através do sistema.

Para cadastrar o novo analista, o administrador acessa o item de menu Usuários e o

sistema exibe uma listagem dos usuários já cadastrados, como pode ser visto na figura 22.

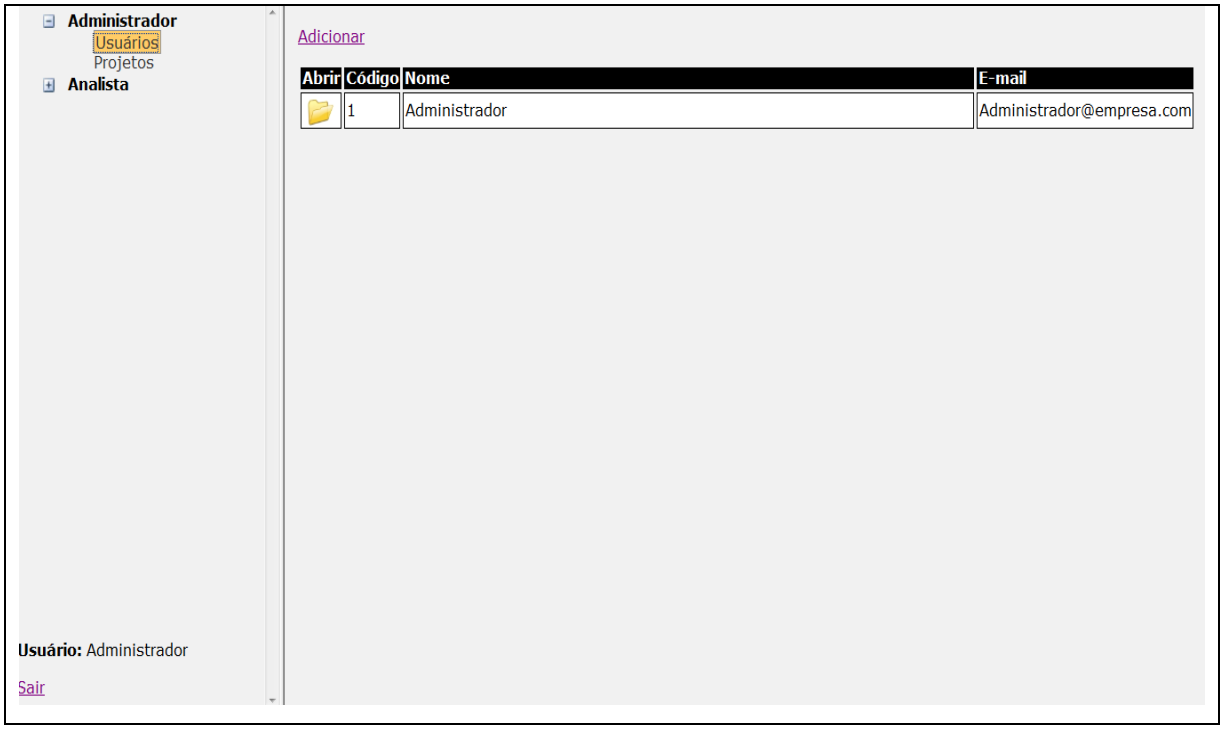


Figura 22 - Tela com a listagem dos usuários

Caso o administrador queira alterar algum dado de algum usuário já cadastrado, basta clicar sobre o ícone em forma de pasta na coluna Abrir do respectivo registro. Clicando no *link* adicionar que aparece acima da listagem, é aberta a tela de cadastro de usuários, onde o administrador informa os dados do analista, conforme exibido na figura 23.

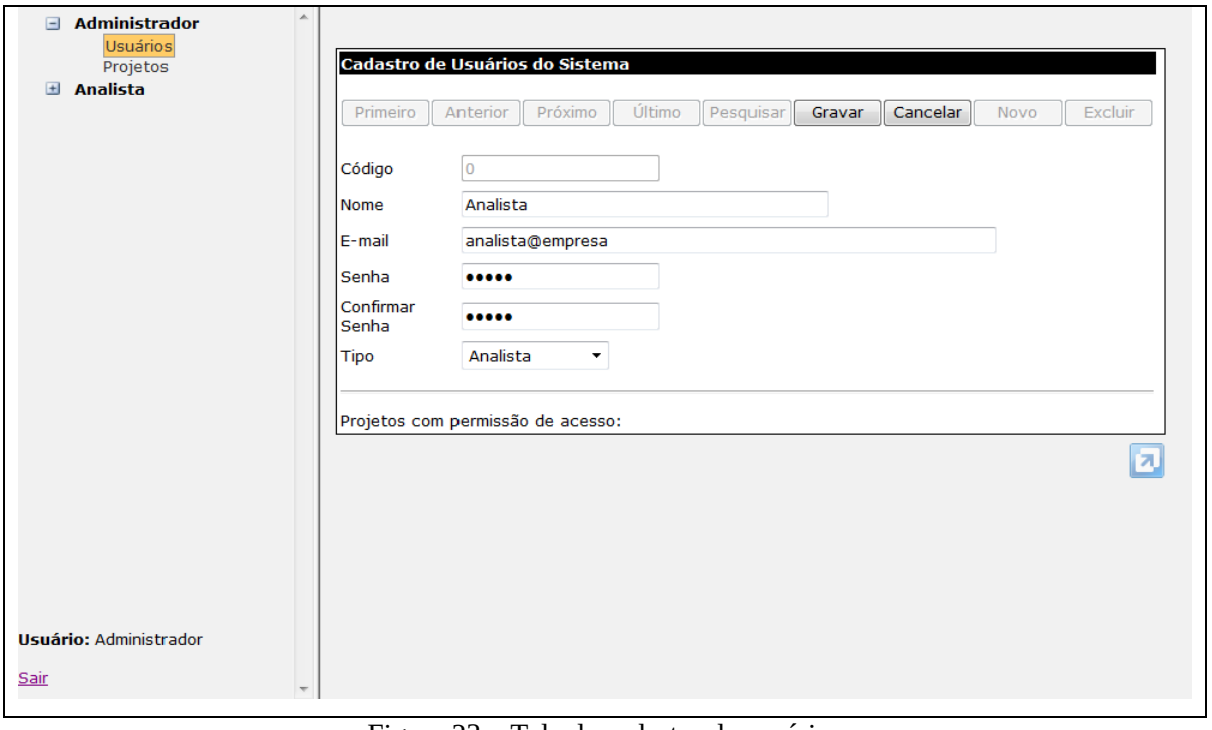


Figura 23 – Tela de cadastro de usuários

Após informar os dados na tela de cadastro, basta clicar no botão Gravar para que o

novo usuário seja inserido no banco de dados. Dois campos merecem destaque nesta tela. O primeiro é o da senha, onde será definida a senha de acesso do usuário ao sistema. O outro é o campo Tipo, onde será definido se o usuário é um administrador (acesso total ao sistema) ou um analista (acesso às versões em aberto dos projetos aos quais esteja relacionado).

Em seguida, o administrador clica no item de menu Projetos para cadastrar o novo projeto. Ao clicar neste item de menu, também é exibida uma listagem dos projetos já cadastrados e, para incluir outro, o administrador clica sobre o *link* Adicionar para que seja exibida a tela de cadastro de projetos, onde os dados do novo projeto são inseridos, sendo possível, inclusive, definir neste momento quais usuários terão acesso ao projeto, bastando para tanto marcar o *check* com os nomes dos usuários desejados. Essa tela de cadastro é exibida na figura 24.

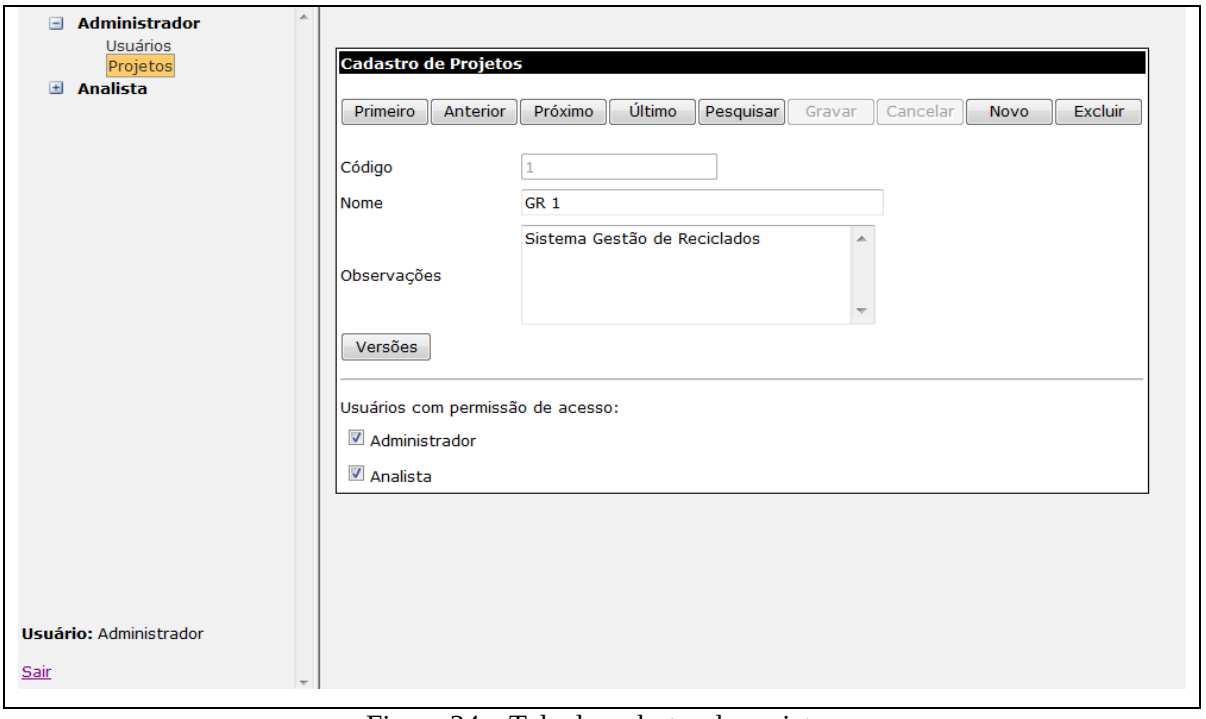


Figura 24 – Tela de cadastro de projetos

Com frequência ao longo do seu ciclo de vida um projeto de banco de dados pode ser alterado inúmeras vezes, seja para modificar requisitos e funcionalidades existentes ou para adicionar funcionalidades ao software modelado, o sistema permite o cadastro de versões para cada projeto, registrar o histórico de alterações. Sendo assim, todas as informações relativas às tabelas, colunas, relacionamentos e diagramas são armazenadas pelo sistema no nível de versões. Para cadastrar uma versão do projeto para que o analista possa trabalhar com ela, o administrador clica no botão Versões que está visível na tela de cadastro do projeto, se o projeto estiver gravado. O sistema apresentará uma tela de listagem, exibindo as versões

cadastradas para o projeto em questão, sendo que para incluir uma versão basta clicar no *link* Adicionar, o que fará com que o sistema exiba a tela de cadastro de versões, ilustrada na figura 25.

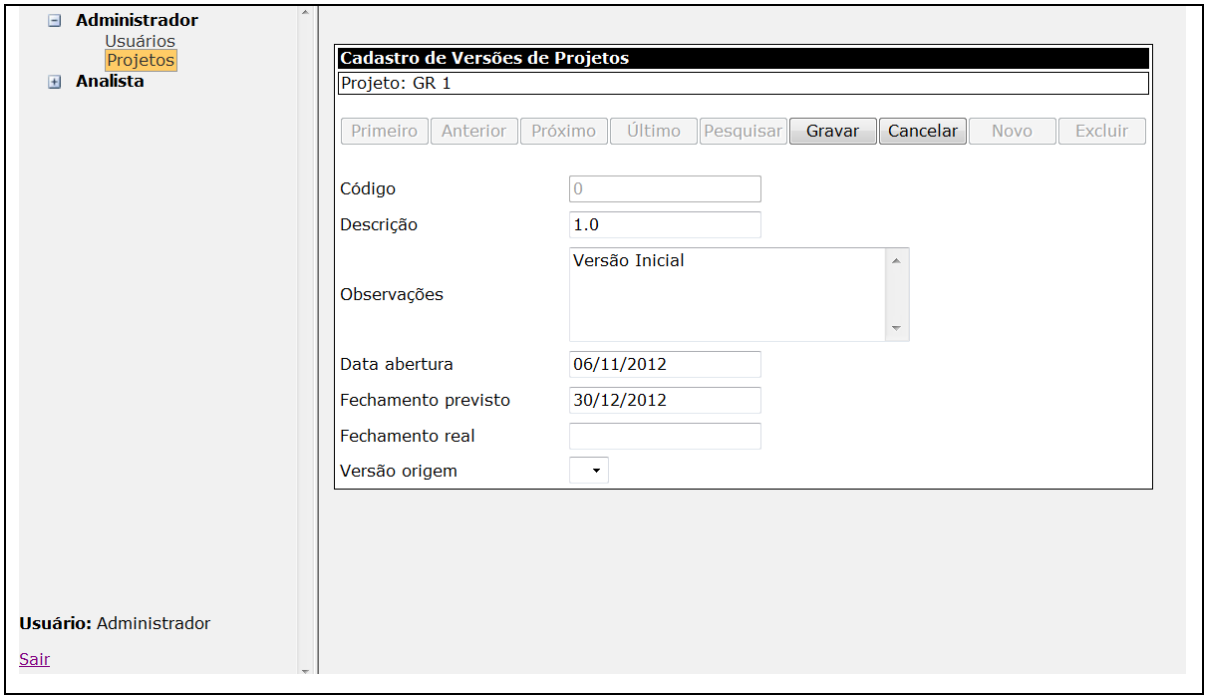


Figura 25 – Tela de cadastro de versões

Ao cadastrar uma versão, é necessário que o administrador informe as datas de abertura e fechamento previsto desta versão, pois, isto limita o acesso do analista a data de abertura da versão preenchida. A data de fechamento previsto é apenas informativa, não há nenhuma consistência feita com ela. Para bloquear o acesso a versão, basta o administrador informar a data de fechamento real. Ao criar uma versão, pode-se ainda informar uma versão de origem, sendo que nesse caso a nova versão é criada como uma cópia da anterior.

Após este passo o administrador efetuou todos os cadastramentos necessários para que o analista possa começar a utilizar o sistema, bastando apenas que o analista acesse o sistema informando o seu código de usuário e sua senha. A tela principal para o analista, a qual pode ser vista na figura 26, é mais restrita que a do administrador, já que o analista não tem acesso ao cadastro de projetos, versões e de usuários.

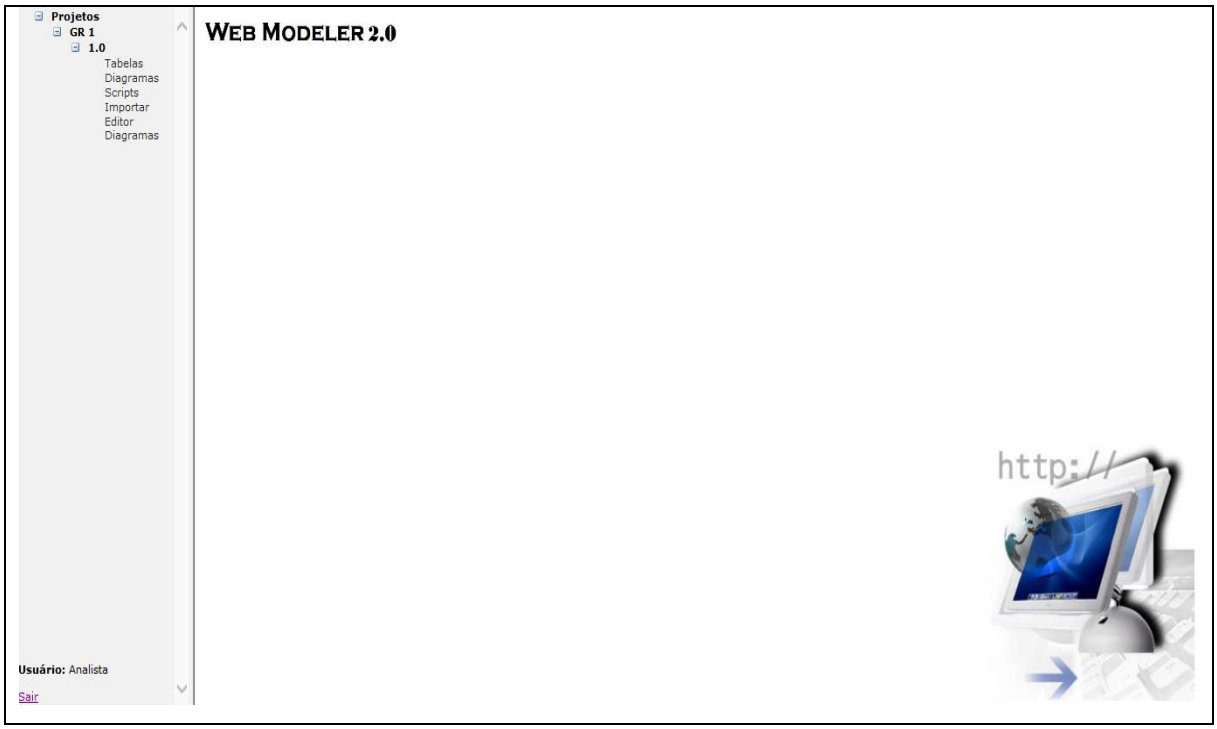


Figura 26 – Tela principal para o analista

O menu que aparece para os analistas traz todos os projetos aos quais os mesmos têm acesso e, para cada projeto, todas as versões que estão em aberto, isto compreende as versão que o analista pode fazer alterações.

O item de menu Diagramas acessa o cadastro de diagramas, que permite a modelagem de um diagrama. Para a atualização deste diagrama no momento da importação do modelo de um banco de dados, é necessário que este diagrama seja informado. Para criar o diagrama deve ser clicado no item de menu Diagramas e informar os dados do diagrama inicial. A tela de cadastro de diagramas pode ser visualizada na figura 27.

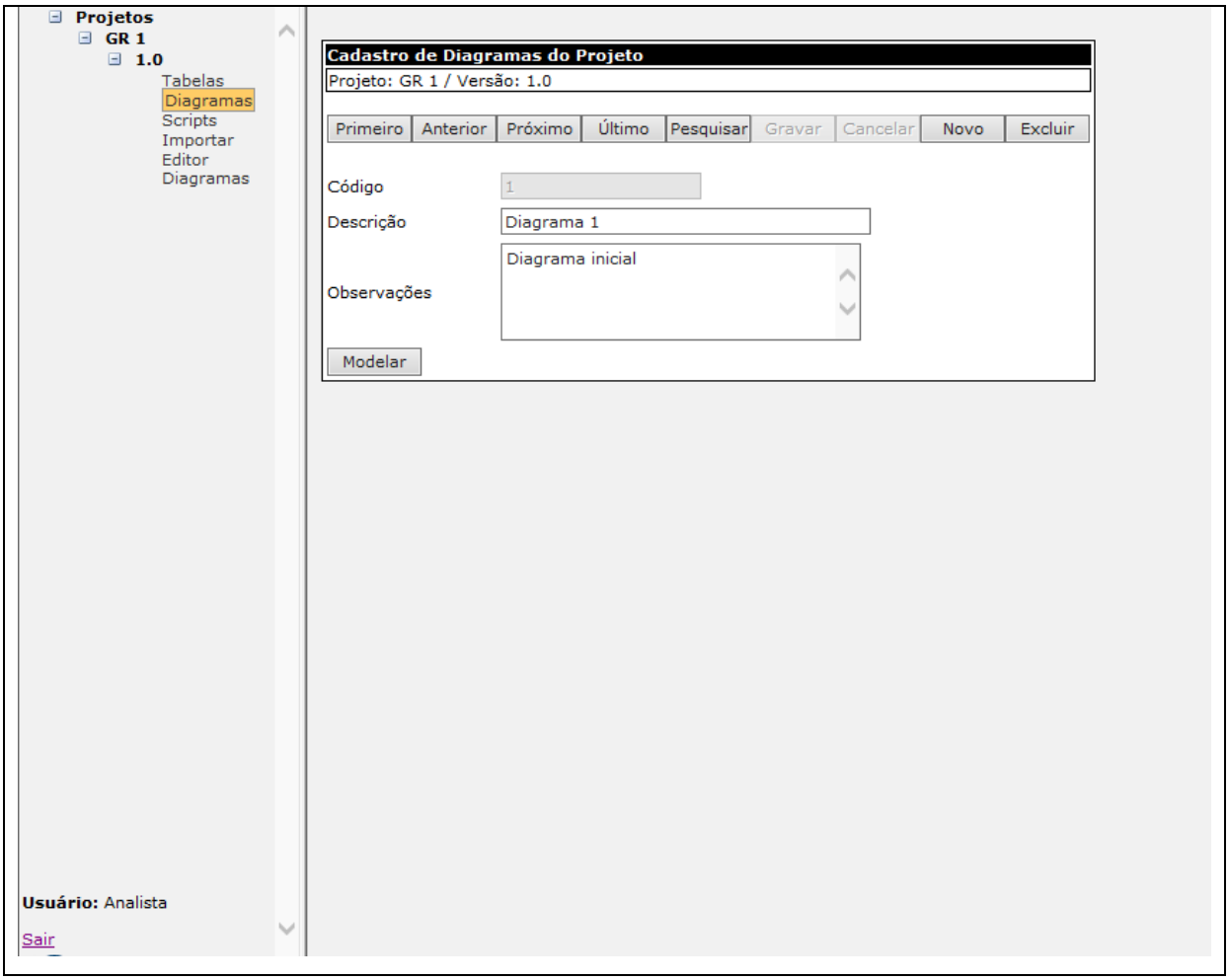


Figura 27 – Tela de cadastro de diagramas

O item de menu Importar acessa à tela para importar as tabelas e um banco de dados existentes em uma versão de um projeto, a tabela importada aparece na listagem de tabelas que segue o mesmo padrão das demais listagens do sistema, isto é, após importada, altera-se uma tabela clicando no ícone da coluna Abrir ou pode-se criar uma tabela acionando o item de menu Adicionar. A tela de importação de tabelas pode ser vista na figura 28.

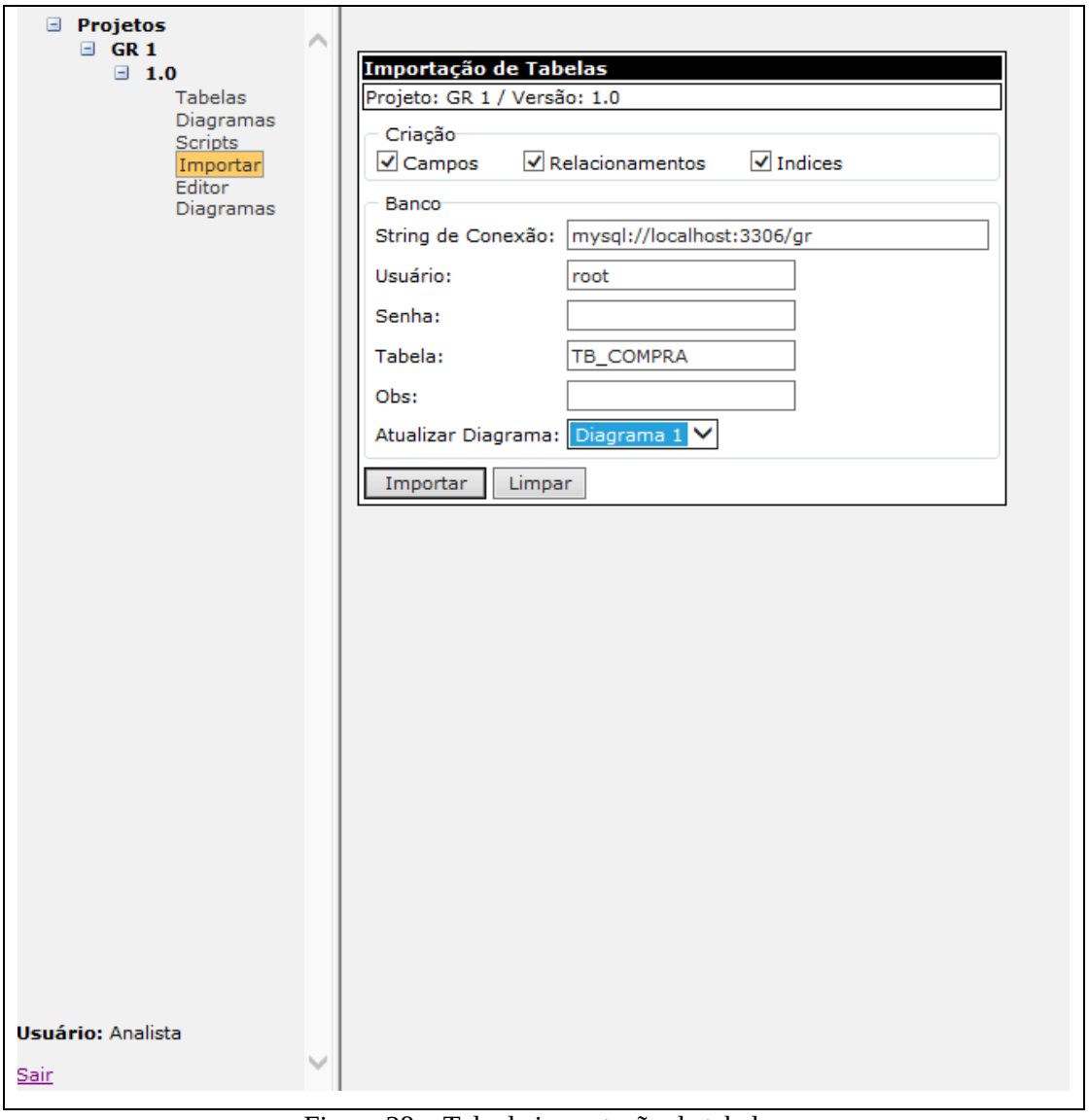


Figura 28 – Tela de importação de tabelas

Na parte superior desta tela, existe o campo criação onde devem ser marcados os itens que se deseja importar de um banco de dados. Para conexão com o banco de dados é necessário informar a *string* de conexão, o usuário e senha. Além destas informações, é necessário que seja informado o nome da tabela e se deseja atualizar um diagrama se o mesmo já existir no projeto. Após a importação da tabela a mesma é listada na listagem de tabelas, a figura 29 demonstra a edição da tabela importada.

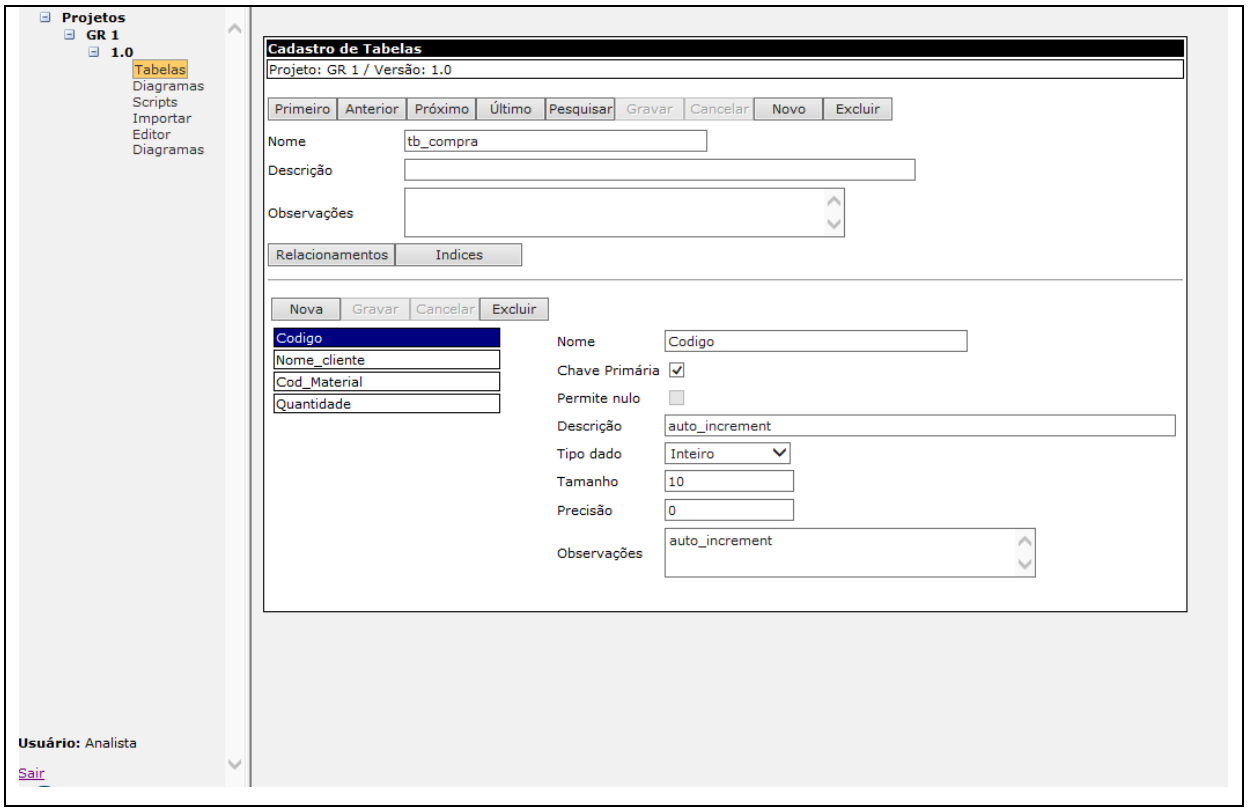


Figura 29 – Tela de edição de tabelas

Se a opção de atualizar diagrama for marcada na importação de tabelas o sistema gera um diagrama inicial do banco de dados, onde este diagrama pode ser alterado. A figura 30 mostra o diagrama inicial gerado pelo sistema de importação.

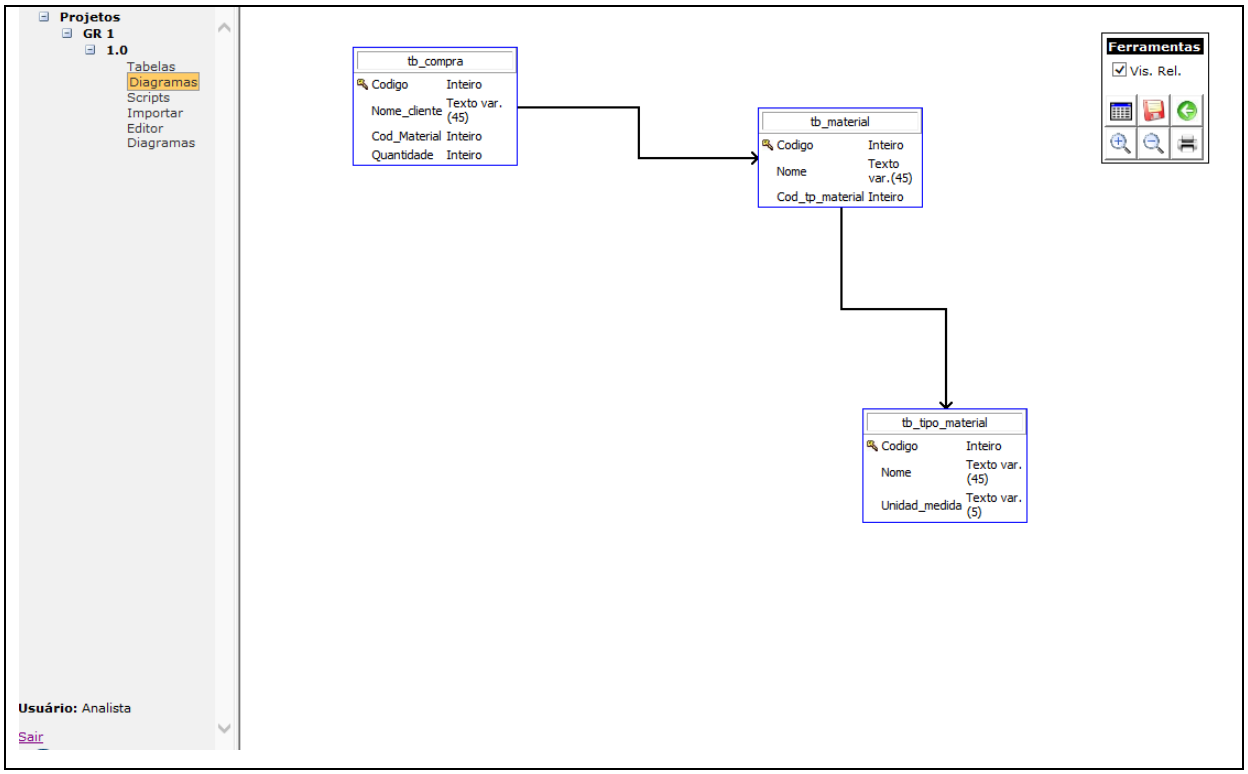


Figura 30 – Tela de edição de tabelas

Após concluir a importação e atualização da modelagem da versão do projeto de banco de dados, o analista pode gerar os arquivos com os comandos SQL necessários à criação do banco de dados, através da opção de menu Scripts. Esta opção faz com que o sistema exiba a tela que pode ser vista na figura 31.

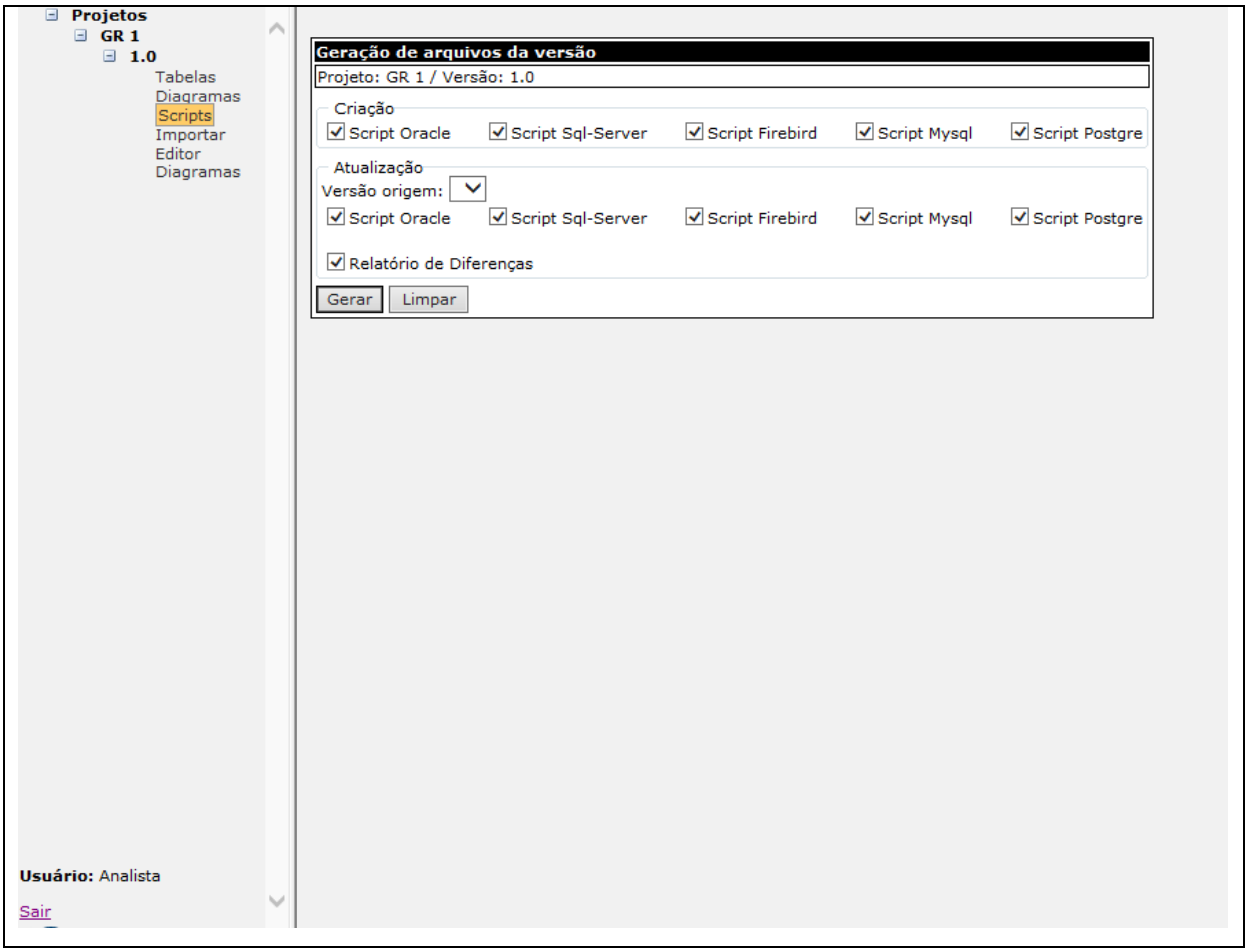


Figura 31 – Tela de geração de *scripts*

Nesta tela, o analista define os SGBD para os quais deverá ser feita a geração dos *scripts* de criação do banco de dados e, se for o caso, também os *scripts* de atualização a partir de uma versão anterior, bem como o relatório de diferenças entre estas versões. Tanto os *scripts* quanto o relatório são apresentados ao usuário no *browser* na forma de um arquivo PDF único, que pode ser visto nos anexos deste trabalho. Em princípio, a geração deste arquivo encerra o processo de criação de uma versão para um banco de dados, sendo que o analista ainda pode criar os diagrama de caso de uso, de classes e de atividades, após a criação dos diagramas a versão pode ser encerrada, o passo seguinte é o fechamento da versão por parte do administrador do sistema, que deve fazê-lo na tela de cadastro da respectiva versão.

Para modelar os diagramas UML, o analista clica no item Editor Diagramas no menu do sistema. Este item abre um editor com a possibilidade de criar a modelagem de diagramas de classe, de casos de uso e de atividades. A figura 32 mostra a tela do editor de diagramas do

WebModeler 2.0.

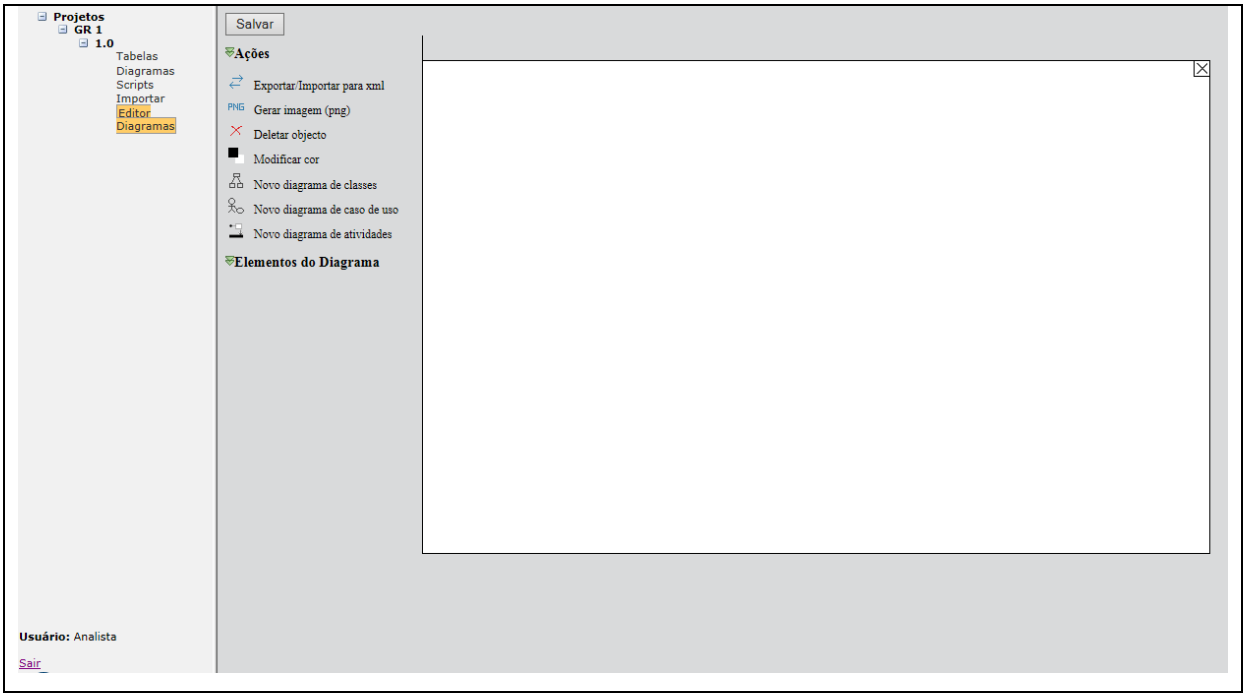


Figura 32 – Tela de edição de diagramas UML

Para modelar um novo diagrama, o analista clica sobre o diagrama desejado que está disponível no menu do editor. A figura 33 mostra a tela de edição com o diagrama de classes sendo modelado. No pacote `modulo.*`, pode ser visto as classes `compra e produto`, com seus métodos definidos.

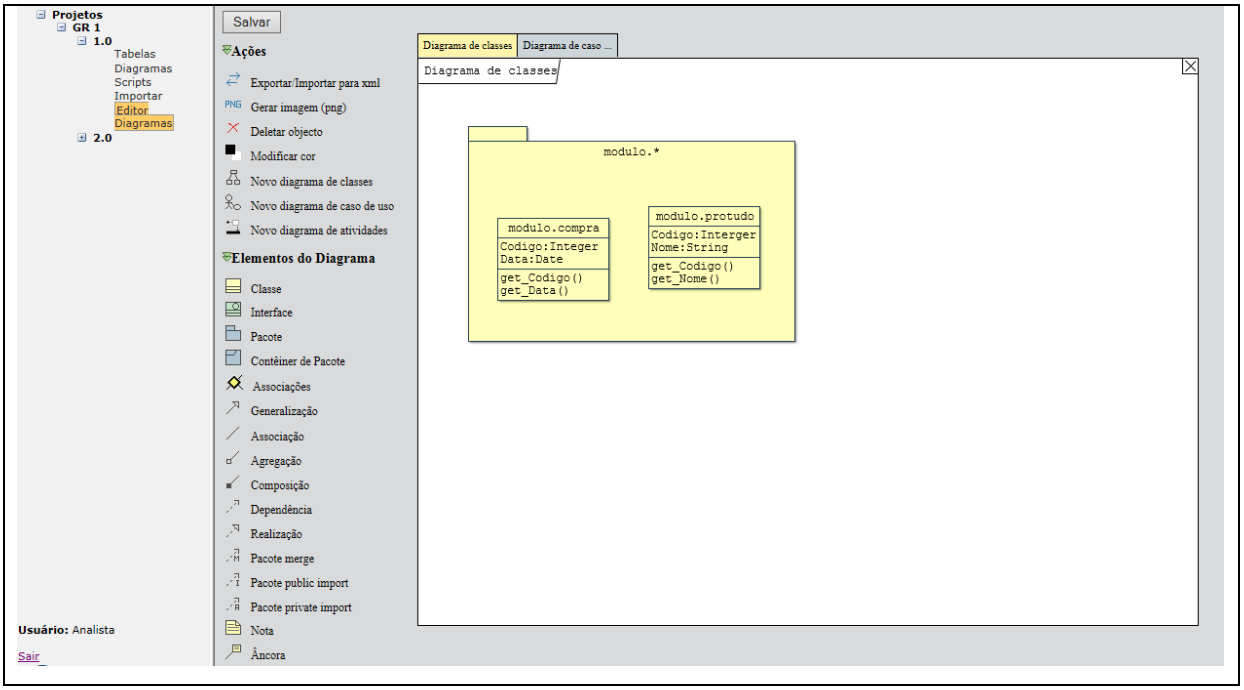


Figura 33 – Tela de edição do diagrama de classes

A figura 34 mostra a tela do editor de diagramas com a modelagem de um diagrama de casos de uso.

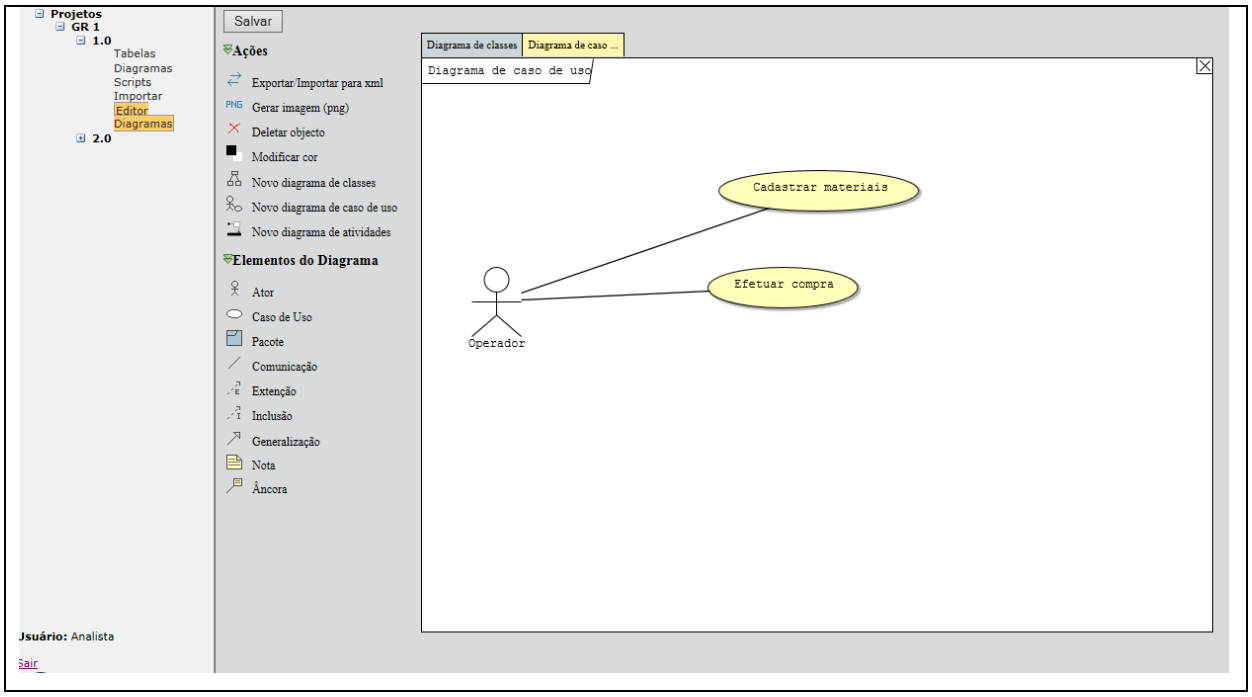


Figura 34 – Tela de edição do diagrama de casos de uso

A figura 35 mostra a tela de edição do diagrama de atividades.

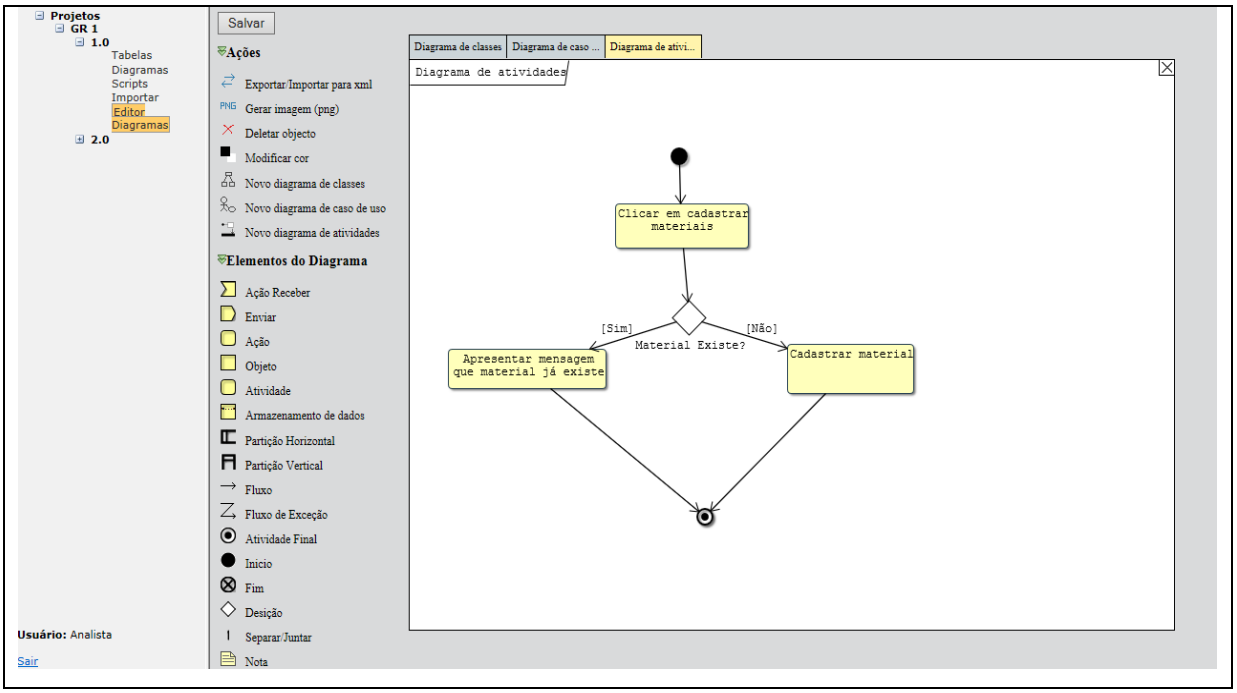


Figura 35 – Tela de edição do diagrama de atividades

No apêndice A constam, o *script* gerado para a criação do banco de dados no SGBD Mysql. Já o apêndice B traz o *script* de atualização para a versão 2.0, os demais banco geram scripts muito idênticos, por este motivo foi incluído somente um SGBD neste trabalho. O apêndice C traz o relatório de diferenças entre as versões 1.0 (cujos passos da modelagem

forem exibidos até aqui) e 2.0, sendo que esta última foi criada posteriormente com base na anterior, mas com as seguintes alterações:

a) inclusão da coluna `Data_compra` (do tipo data) na tabela `TB_COMPRA`;
b) exclusão da coluna `Unidad_medida` da tabela `TB_TIPO_MATERIAL`;
c) inclusão da tabela `TB_UNIDAD_MEDIDA`, tendo as colunas `Código` (do tipo inteiro longo), `Descricao` (do tipo texto variável com 8 posições), `Preco` (do tipo numérico, com 11 posições e precisão igual a 2).

3.4 RESULTADOS E DISCUSSÃO

O software apresentado neste trabalho permite a modelagem de bancos de dados relacionais no ambiente web, possibilitando o cadastramento de projetos e suas versões, bem como as tabelas e relacionamentos existentes em cada versão. Essa modelagem pode ser feita, inclusive, de forma gráfica ou então importando a estrutura de tabelas, relacionamentos e índices, como ocorre em diversas ferramentas CASE disponíveis no mercado. É possível ainda gerar *scripts* para a criação do banco de dados nos SGBD Oracle, MSSqlServer, Mysql, PostgreSql e Firebird e, além disso, graças ao controle de versões, também é possível gerar *scripts* de atualização para estes bancos de dados. Para a modelagem de diagramas da UML o software apresentado disponibiliza a geração de diagramas de casos de uso, de classes e atividades. Esta modelagem pode ser feita de forma gráfica, seguindo a linha das ferramentas CASE disponíveis no mercado.

Com relação às ferramentas CASE existentes no mercado, o software desenvolvido tem algumas limitações, principalmente na questão da modelagem gráfica, pois não há recursos para formatação das tabelas e relacionamentos. Muito da limitação da modelagem gráfica do WebModeler 2.0 pode ser atribuído a atual falta de recursos dos *browsers* e do JavaScript em se tratando de criação de elementos gráficos (retas, figuras geométricas, etc.). Além disso, a falta de experiência com desenvolvimento web também contribui para o desenvolvimento de recursos mais avançados no aplicativo. Em contrapartida, o WebModeler 2.0 oferece vantagens como o acesso remoto sem a necessidade de instalação local (por tratar-se de um aplicativo web), controle de projetos e versões, cadastro de usuários e restrições de acesso por meio do relacionamento entre projetos e usuários.

As figuras 36 e 37 foram incluídas a seguir para ilustrar a semelhança na modelagem

de um banco de dados no software ora apresentado e numa ferramenta CASE já existente (no caso, o PowerDesigner). O mesmo banco de dados foi modelado em ambos os aplicativos.

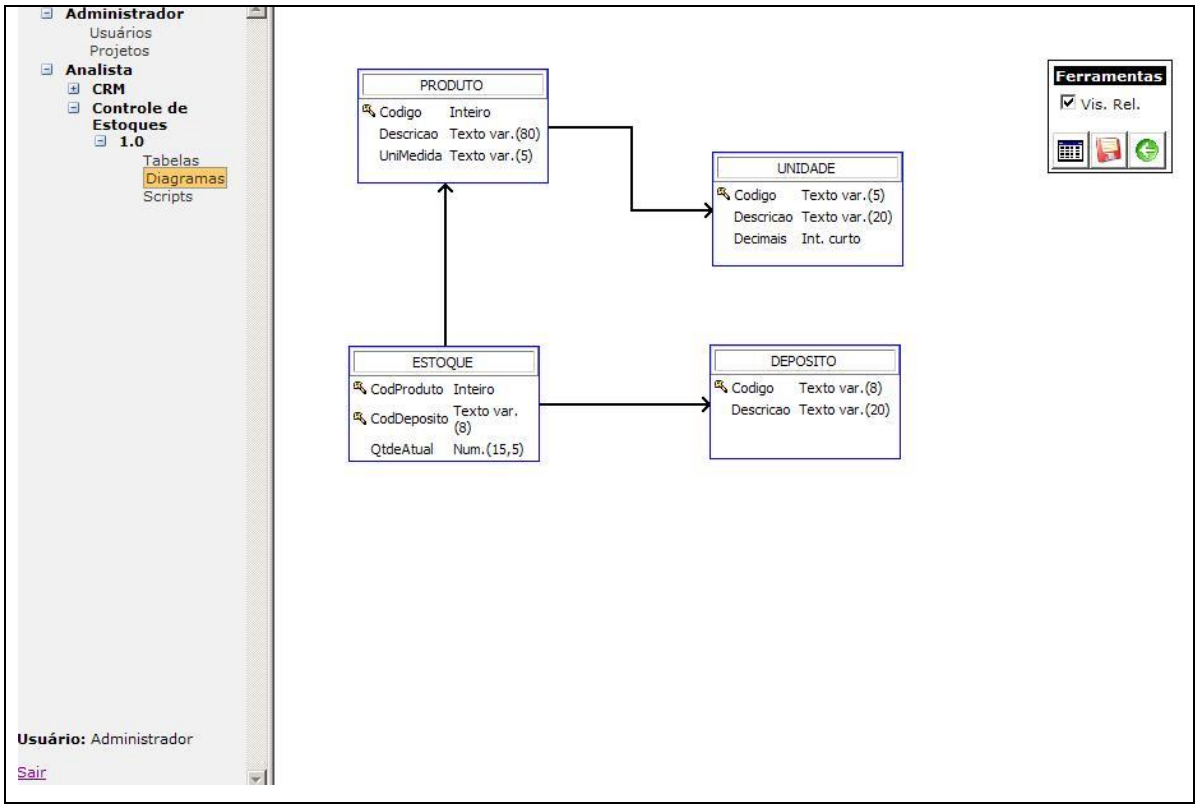


Figura 36 – Modelagem no WebModeler 2.0

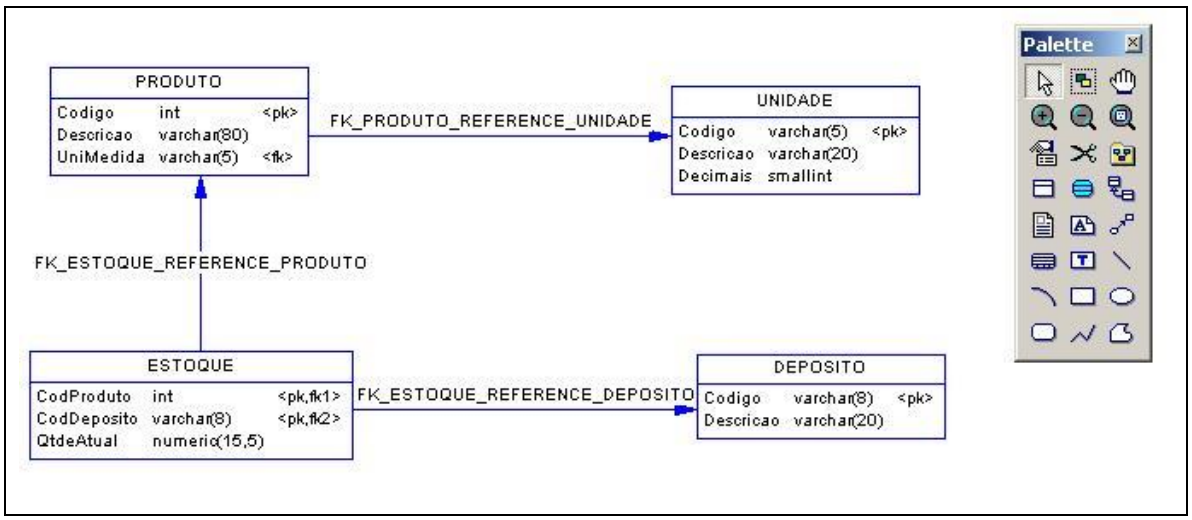


Figura 37 – Modelagem no PowerDesigner

As figuras 38 e 39 foram incluídas a seguir para ilustrar a semelhança nos diagramas UML no software ora apresentado e numa ferramenta CASE já existente (no caso, o Enterprise Architect). O mesmo diagrama foi modelado em ambos sistemas.

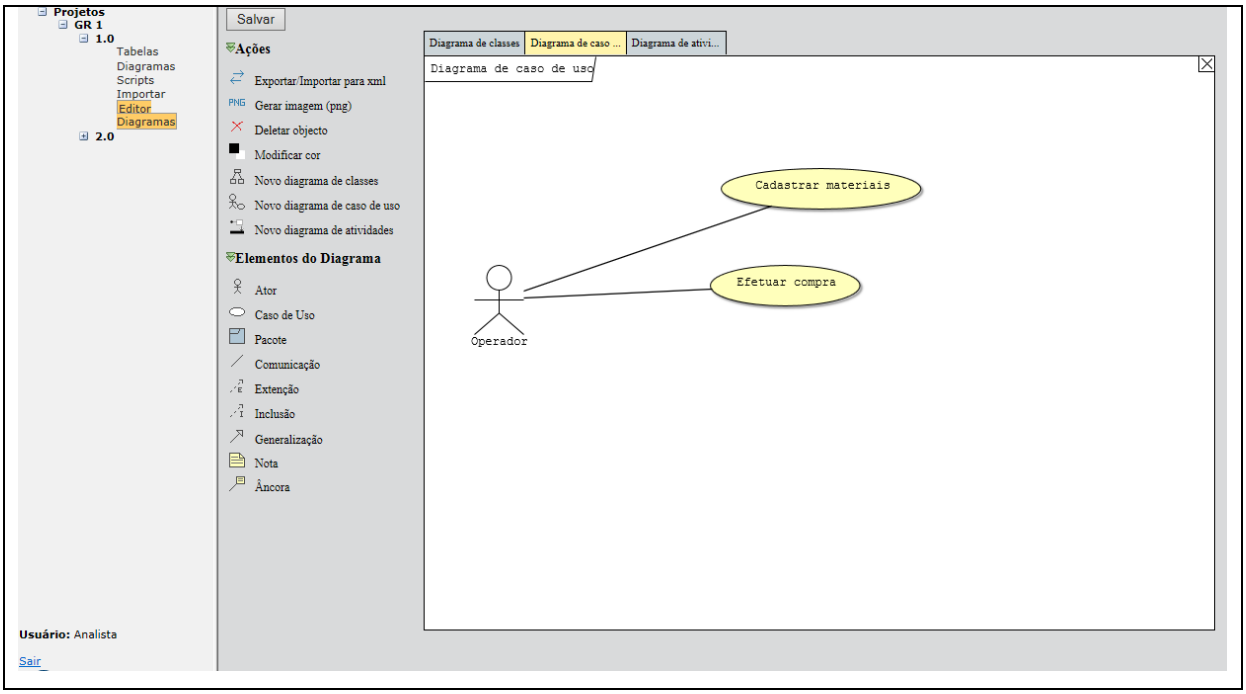


Figura 38 – Modelagem UML no WebModeler 2.0

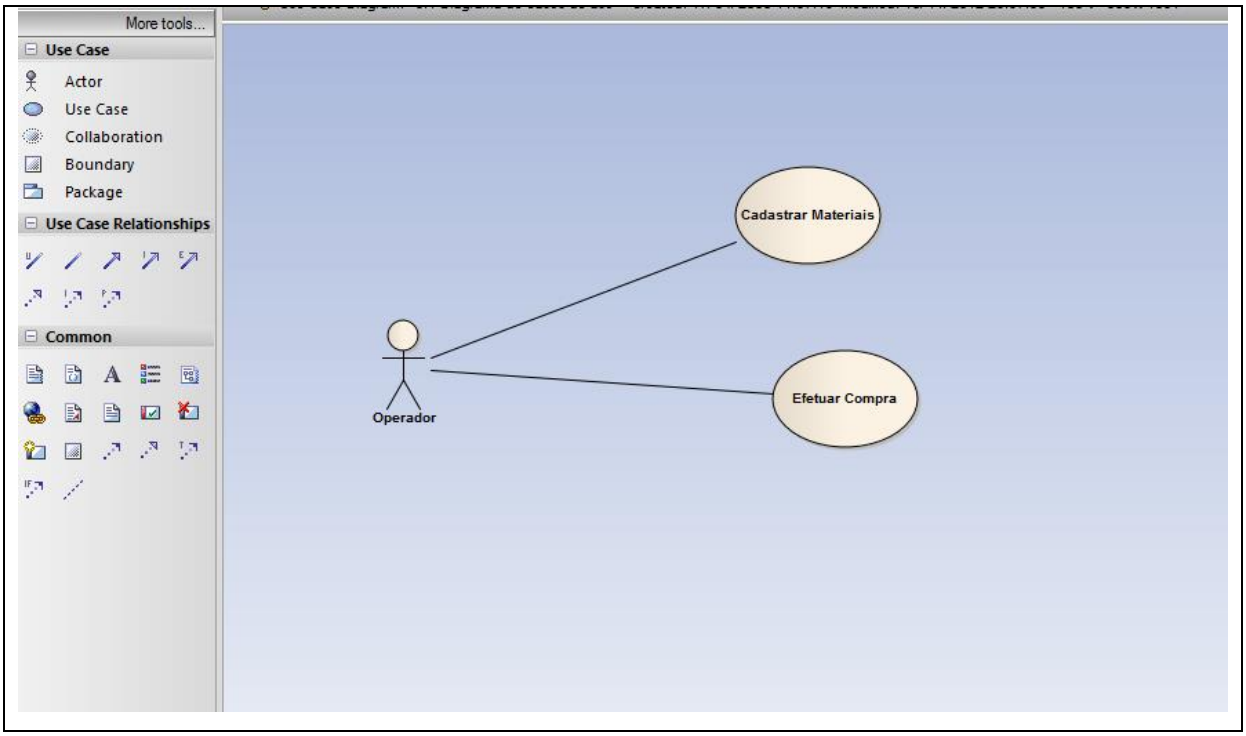


Figura 39 – Modelagem no Enterprise Architect

As figuras 40 e 41, comparam a modelagem de diagramas de classes no WebModeler 2.0 e Enterprise Architect.

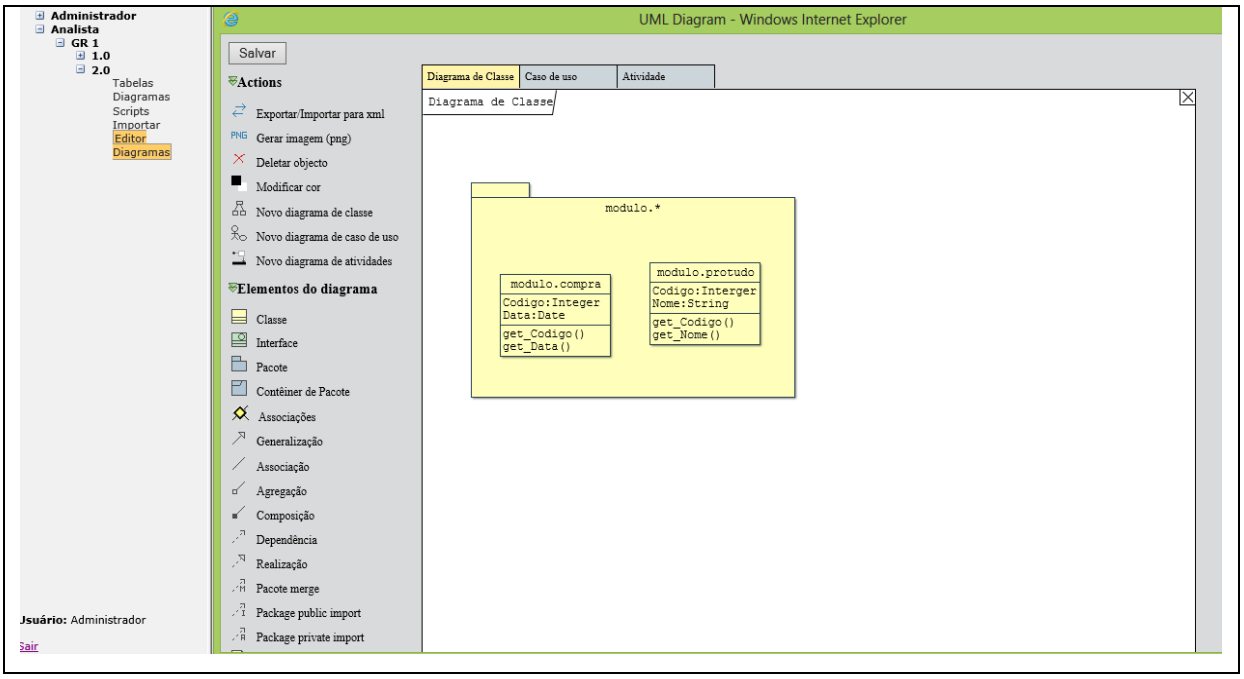


Figura 40 – Modelagem diagrama de classes no WebModeler 2.0

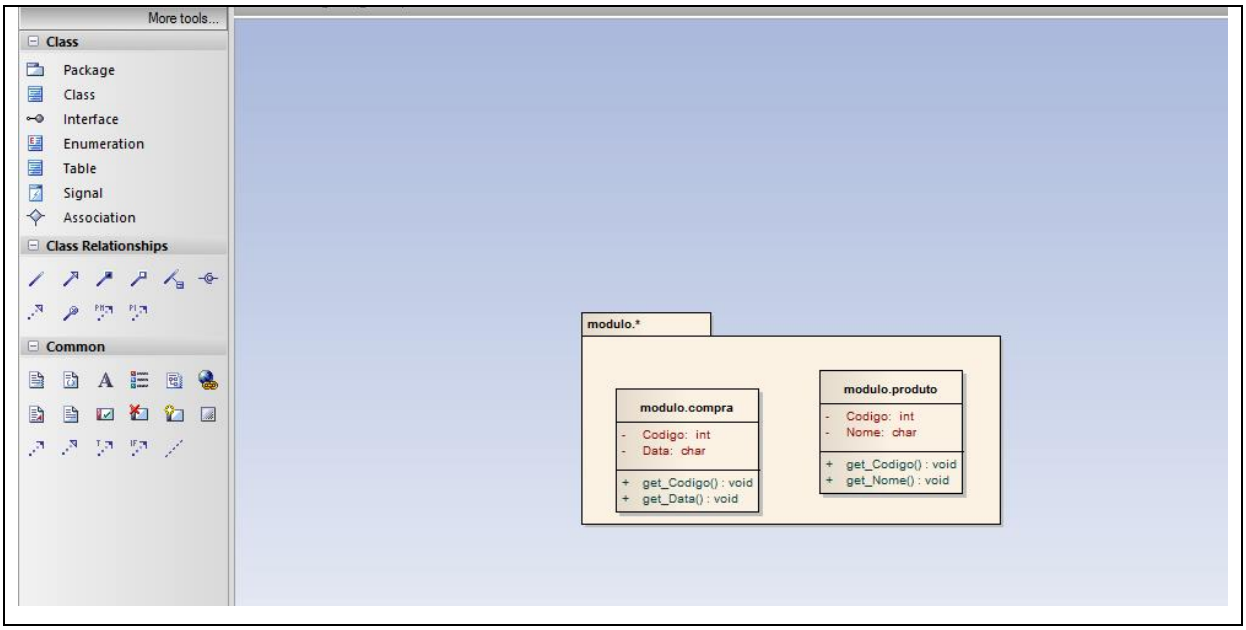


Figura 41 – Modelagem diagrama de classes no Enterprise Architect

As figuras 42 e 43, comparam a modelagem de diagramas de atividades no WebModeler 2.0 e Enterprise Architect.

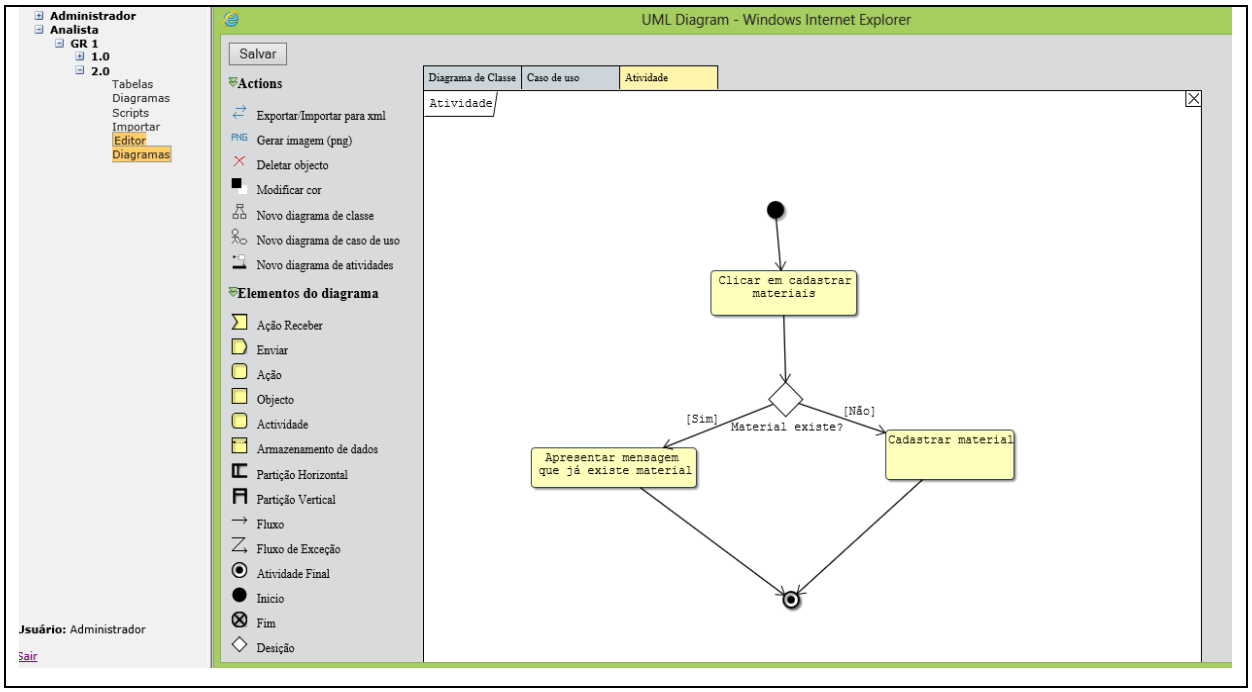


Figura 42 – Modelagem diagrama de atividades no WebModeler 2.0

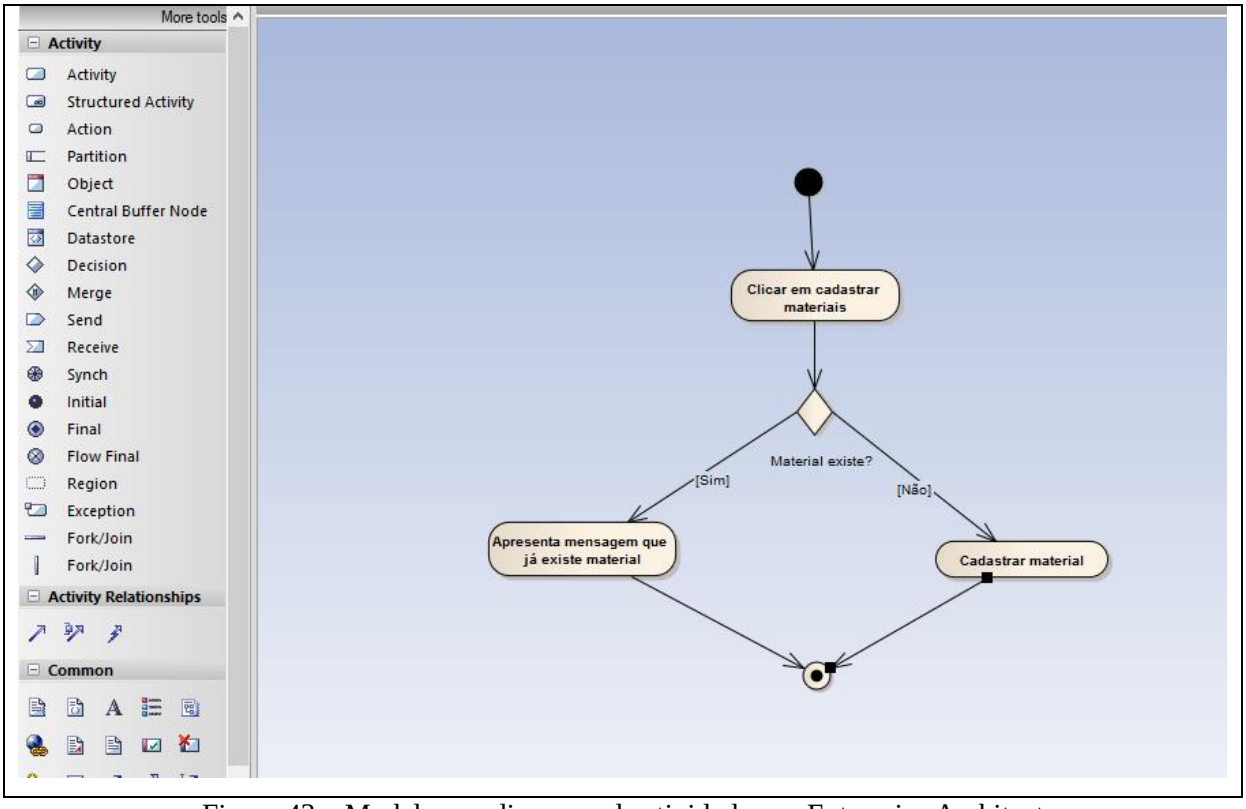


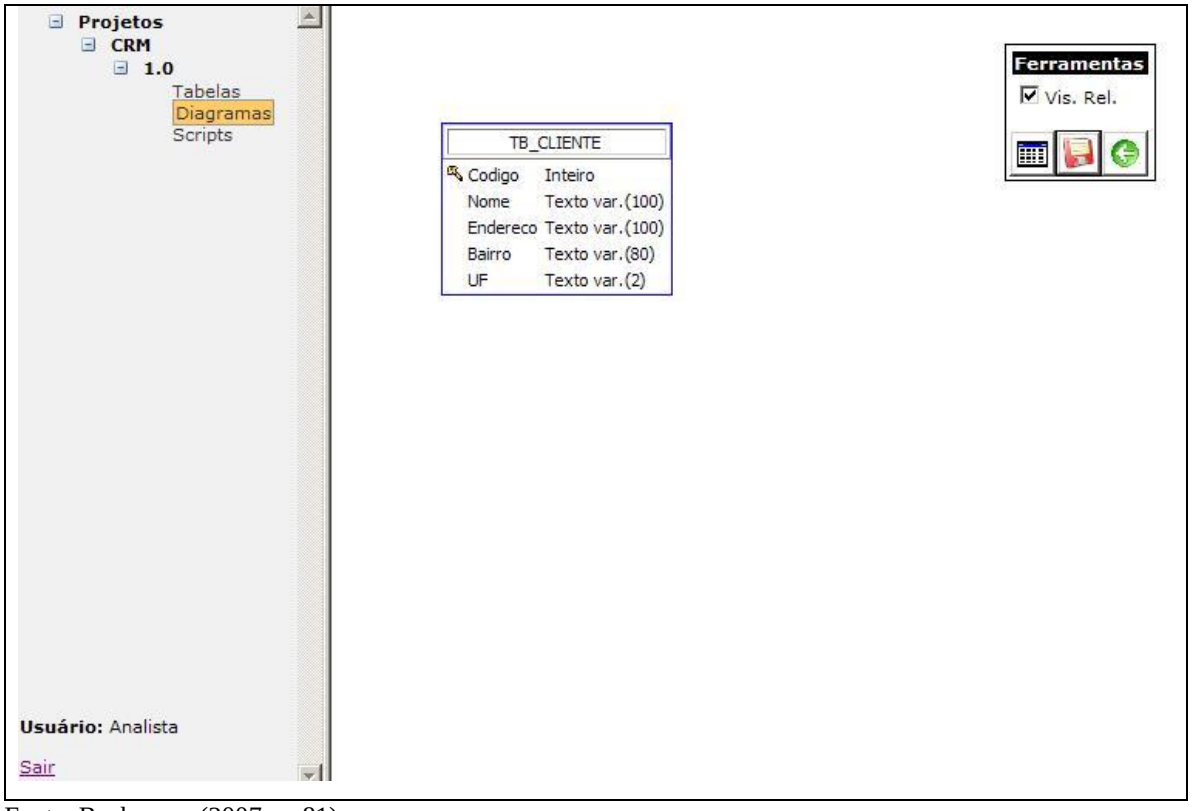
Figura 43 – Modelagem diagrama de atividades no Enterprise Architect

Já com relação ao trabalho de Bernardi (1997), o software atual apresenta inúmeros novos recursos. Pode-se citar também que o trabalho atual permite salvar os diagramas desenvolvidos, algo que não era possível no trabalho anterior, além de ser um aplicativo web,

ao invés de *desktop*.

Com relação ao trabalho de Cândido (2007), o WebModeler 2.0 apresenta uma interface mais amigável, facilitando a utilização no meio acadêmico. Também pode-se citar que o WebModeler 2.0 gera os scripts para criação e atualização do banco de dados o software de Cândido (2007) não tinha esta funcionalidade.

Já com relação ao WebModeler de Bachmann (2007), o WebModeler 2.0 traz novas funcionalidades, como a possibilidade de importar o modelo de um banco de dados, o módulo de modelagem de diagramas também recebeu novas funcionalidades como impressão, aproximação e afastamento de diagramas. A figura 44 exibe a tela de modelagem de diagramas do WebModeler de Bachmann (2007), já a figura 45 mostra a mesma tela no WebModeler 2.0.

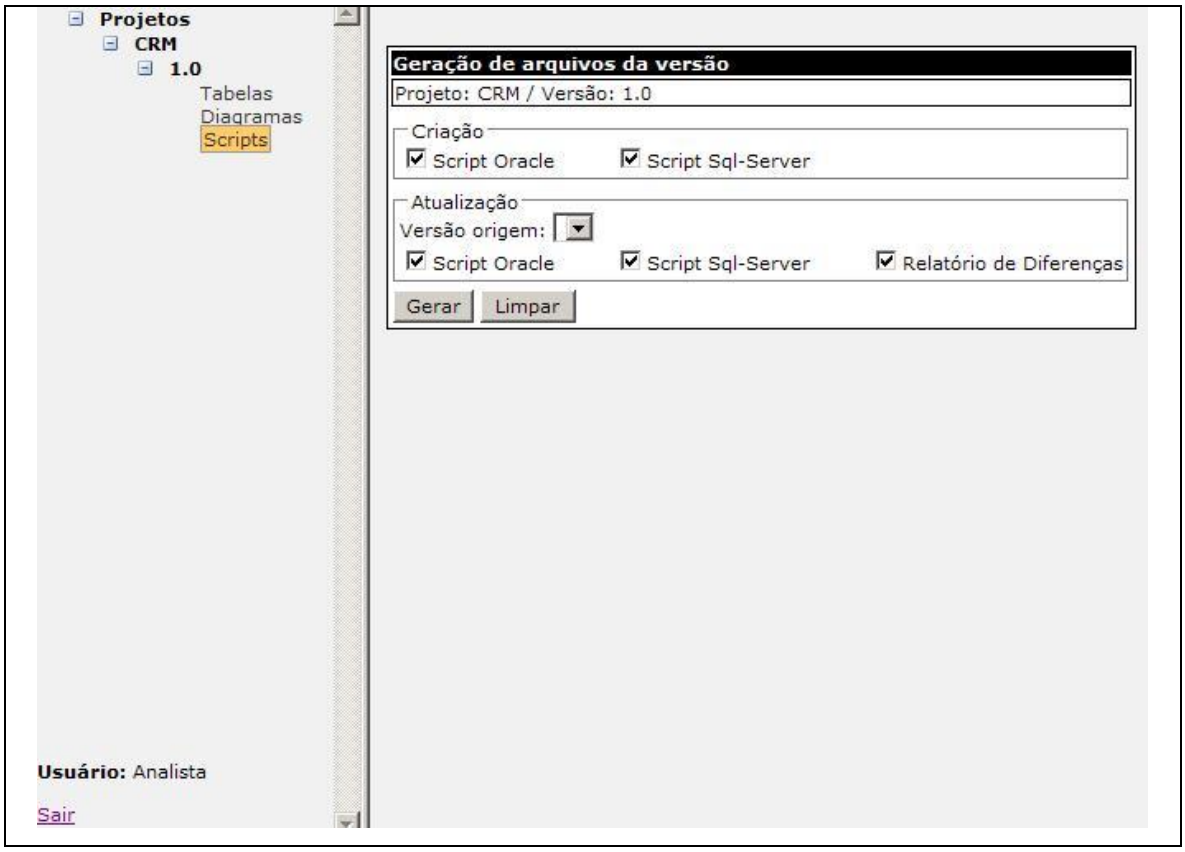


Fonte: Bachmann (2007, p. 81).

Figura 44 – Tela de modelagem gráfica com a tabela definida sistema WebModeler

Figura 45 – Tela de modelagem gráfica com a tabela definida sistema WebModeler 2.0

Outro item que o WebModeler 2.0 se diferencia do software de Bachmann (2007), é a geração de script para os bancos Mysql, PostgreSql e Firebird, além dos bancos Oracle e MysqlServer que WebModeler já disponibilizava. Na figura 46 pode ser observada a tela de geração de script de WebModeler de Bachmann (2007), já a figura 47 mostra a tela de WebModeler 2.0.



Fonte: Bachmann (2007, p. 82).

Figura 46 – Tela de geração de *scripts* WebModeler

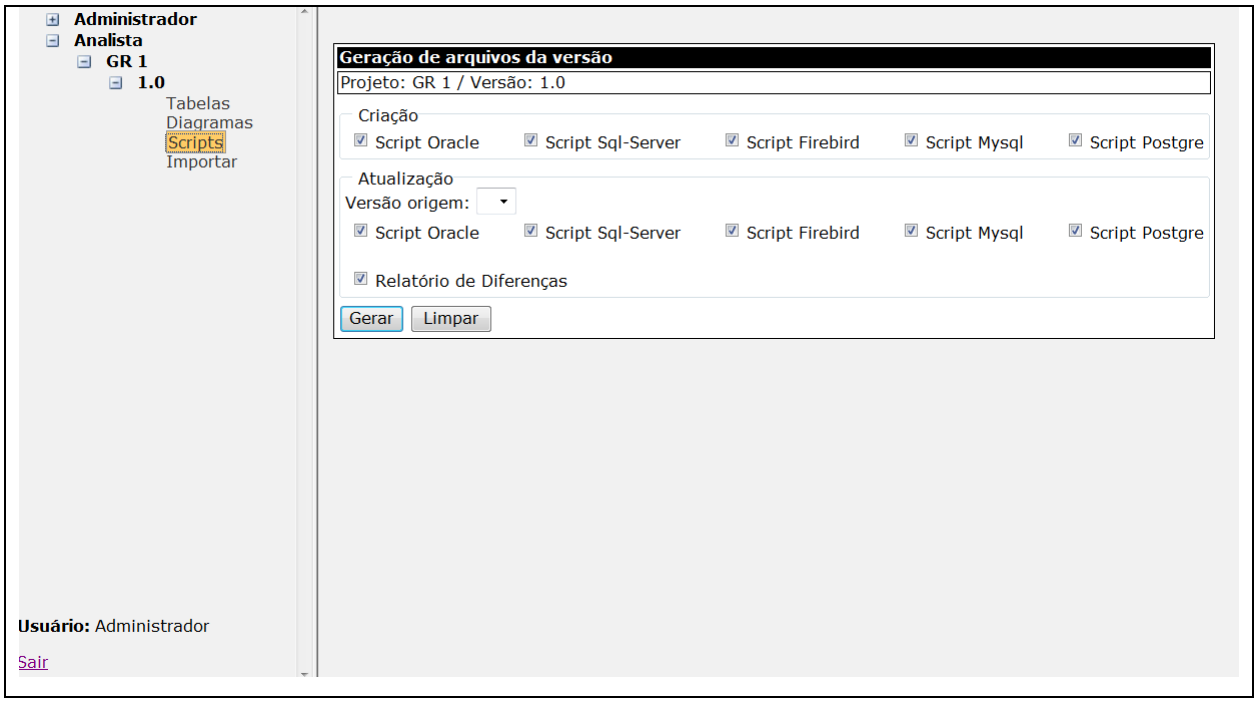


Figura 47 – Tela de geração de *scripts* WebModeler 2.0

Em termos de desenvolvimento de novas funcionalidades, o WebModeler 2.0 disponibilizou o cadastro de índices para banco de dados, o módulo de importação de modelo de banco de dados Mysql e o módulo para criação de diagramas da UML. Este que já abre

uma nova linha de funcionalidades que não se restringe apenas na modelagem para banco de dados.

Em termos de desenvolvimento do aplicativo, as tecnologias, técnicas e ferramentas empregadas mostraram-se bastante eficientes. A utilização do AJAX início do desenvolvimento pareceu um pouco complicada, devido a pouca prática com esta técina, porém durante o decorrer do trabalho esta dificuldade diminuiu. Outro ponto que dificultou a realização deste trabalho foi a implementação de novas funcionalidades em HTML5 e CSS3, pois o trabalho de Bachmann (2007) foi desenvolvido para o *brower* Internet Explorer 6 (IE), que não tem compatibilidade com esta tecnologica. Devido a esta incompatibilidade foram obrigados ajustes no WebModeler, para o funcionamento em *browser* IE mais atual. No WebModeler de Bachmann (2007) utilizou HTML Padrão e Javascript para modelagem de banco de dados, no presente trabalho foram utilizados recursos do HTML5 e CSS3, para melhorar os gráficos.

4 CONCLUSÕES

Ao fim deste trabalho pode-se concluir que os objetivos iniciais foram alcançados. O aplicativo web desenvolvido permite que um banco de dados relacional seja modelado, inclusive através de diagramas, de forma independente do SGBD onde será criado (Oracle, MSSqlServer, Mysql, PostgreSql, Firebird). Os tipos de dados para estas colunas são próprias do sistema e não nativas de algum SGBD específico e com a possibilidade se conectar a um banco de dados Mysql e realizar a importação de suas tabelas, relacionamentos e índices, desempenhando assim a funcionalidade de engenharia reversa de modelo de SGBD. Essa independência de algum SGBD específico pode ser considerada importante principalmente porque diversas linguagens de programação, como C# e Java, por exemplo, permitem a geração de programas cujo código-fonte é totalmente compatível com vários SGBDs relacionais. Não é necessária uma preocupação com o SGBD que será utilizado posteriormente e, com a ferramenta desenvolvida neste trabalho, também é possível criar o modelo do banco de dados sem esta preocupação.

Com relação à funcionalidade de engenharia reversa de modelo de SGBD, é possível obter a estrutura, modelo, relacionamentos e índices de um banco da dados Mysql, podendo inclusive gerar automaticamente um diagrama contendo as informações coletadas neste banco de dados.

As vantagens para os usuários do software desenvolvido em relação a outras ferramentas CASE são várias. Em primeiro lugar, trata-se de um aplicativo web que utiliza um banco de dados, ao invés de ser uma aplicação *desktop* onde os modelos são salvos em arquivos binários. Com isso, além de uma maior segurança no armazenamento dos modelos, também é facilitado o acesso de inúmeros usuários ao mesmo projeto, pois todos podem acessá-lo a qualquer hora e de qualquer lugar e trabalhar simultaneamente no mesmo, sem a necessidade de, por exemplo, possuir uma rede compartilhada para disponibilizar o arquivo binário para os usuários envolvidos. Por outro lado o software possui uma pequena limitação, pois não impede que dois ou mais usuários alterem os dados de um mesmo objeto dentro da mesma versão de um projeto, o que pode gerar alguns problemas.

Ainda com relação aos usuários, o software permite, além do cadastro destes, o relacionamento entre projetos e usuários, ou seja, a definição dos analistas que poderão acessar cada projeto, lembrando que o acesso ao sistema só é possível mediante informação de senha.

Outra vantagem deste software é o controle das versões dos projetos, com a possibilidade de se gerar *scripts* de criação e atualização de bases de dados para os SGBD Oracle, MSSqlServer, Mysql, PostgreSql e Firebird, estes sendo os mais comuns SGBD relacionais do mercado atualmente. Também podendo utilizar engenharia reversa com a funcionalidade de importação de modelo de tabelas, relacionamentos e índices do SGBD Mysql, possibilitando a criação de uma documentação inicial a partir do modelo do banco de dados. Outra funcionalidade é o cadastro de índices para os SGBD, que são utilizados para a otimização de consultas a um banco de dados.

Ainda com relação às vantagens, o software permite que os usuários do sistema tenham maior controle do projeto, desenvolvendo os diagramas de casos de uso, de classes e atividades, sendo estes dois integrantes dos diagramas mais comuns da UML.

Quanto ao mercado para utilização da ferramenta desenvolvida, percebeu-se que além dos fins comerciais (empresas desenvolvedoras de software) a mesma pode ser utilizada para fins didáticos, em especial em disciplinas que visam prover conhecimentos básicos a cerca de bancos de dados relacionas, projeto de bancos de dados e modelagem de sistemas.

4.1 EXTENSÕES

Para trabalhos futuros podem ser dadas várias sugestões baseadas no atual. Inicialmente, podem ser melhorados os recursos já existentes, disponibilizando novas ferramentas para modelagem gráfica, como recursos de personalizar os relacionamentos, alterando suas posições e o seu formato. A geração de scripts pode sofrer alterações gerando um arquivo SQL para cada banco de dados, estes arquivos gerados podem ser compactados em um arquivo final, facilitando a utilização dos *scripts*. O módulo importação de modelo de banco de dados também pode sofrer ajustes, melhorando a busca as tabelas a serem importadas e principalmente para permitir o suporte de outros SGBD relacionais disponíveis no mercado, tais como Oracle, Interbase, PostgreSql, Firebird entre outros. O módulo de geração de diagramas UML pode sofrer alguns ajustes, cadastrando os requisitos, expandindo a utilização dos diagramas de casos de uso, que podem ter um relacionamento com os requisitos do sistema, um detalhamento maior do caso de uso pode ser desenvolvido, já o diagrama de classes pode também sofrer ajustes, disponibilizando um recurso para geração de um código fonte inicial a partir do diagrama.

REFERÊNCIAS BIBLIOGRÁFICAS

ASLESON, Ryan; SCHUTTA, N. T. **Fundamentos do Ajax**. Rio de Janeiro: Alta Books, 2006.

BATTISTI, Julio. **SQL Server 2000:** administração e desenvolvimento: curso completo. Rio de Janeiro: Acxel Books, 2001.

BERNARDI, Yuri A. **Protótipo de uma ferramenta de entidade relacionamento (ER) para geração de código para banco de dados (Interbase).** 1997. 86 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

CÂNDIDO, Carlos H. **brModelo:** ferramenta freeware voltada para ensino de modelagem em banco de dados relacional.[S.1], 2007. Disponível em: <http://www.sis4.com/brModelo/Default.aspx>. Acesso em: 21 ago. 2012.

COUGO, Paulo. **Modelagem conceitual e projeto de bancos de dados**. Rio de Janeiro: Campus, 1997.

CRANE, Dave; PASCARELLO, Erick; JAMES, Darren. **Ajax em ação.** Tradução Edson Furmankiewicz e Carlos Schafranski. São Paulo: Pearson Prentice Hall, 2007.

DATE, Christopher J. **Introdução a sistemas de bancos de dados**. Rio de Janeiro: Campus, 2000.

FIREBIRD. **Firebird documentation:** data types, [S.1], 2012. Disponível em: < http://dev.mysql.com/doc/refman/5.5/en/data-types.html>. Acesso em: 15 set 2012.

FORTES, Debora. Web 2.0. **Info Exame**, São Paulo, n. 243, p. 44-49, Jun. 2006.

GARRET, Jesse J. **Ajax**: a new approach to web applications 2005. Disponível em: < http://www.adaptivepath.com/publications/essays/archives/000385.php>. Acesso em: 18 set 2012.

HEUSER, Carlos A. **Projeto de banco de dados**. 3. ed. Porto Alegre: Sagra Luzzato, 2000.

KURNIAWAN, Budi. **Java para a web com servlets, JSP e EJB**: um guia do programador para soluções escalonáveis em J2EE. Rio de Janeiro: Ciência Moderna Ltda., 2002.

MYSQL. **Mysql 5.5 reference manual:** Chapter 11. Data types, 2012. Disponível em: < http://dev.mysql.com/doc/refman/5.5/en/data-types.html>. Acesso em: 25 set 2012.

ORACLE. **Oracle database SQL reference 10g release 2:** datatypes. [S.l.], 2005. Disponível em: <http://download-east.oracle.com/docs/cd/B19306_01/server.102/b14200/sql_elements001.htm#i54330/>. Acesso em: 24 maio 2007.

POSTGRE. **PostgreSQL 9.1.6 documentation,** 2012. Disponível em: <http://www.postgresql.org/docs/9.1/static/reference.html>. Acesso em: 20 set 2012.

SALGUERO, Jose R. R. **jsUML2:** a lightweight HTML5/javascript library for UML 2 diagramming, Cordoba [2012]. Disponível em: <http://code.google.com/p/jsuml2/>. Acesso em: 25 set 2012.

APÊNDICE A – *Script* de criação do banco de dados para Mysql

```
*
Script de criação do banco de dados para Mysql
Projeto: GR 1
Versão: 2.0
Data: 19/11/2012 09:12:40
*/

create table tb_compra (
Codigo int not null,
Nome_Cliente varchar(45),
Cod_material int,
Quantidade int,
Data_compra date,
constraint CP_tb_compra primary key (Codigo)) ;

create table tb_material (
Codigo int not null,
Nome varchar(45),
Cod_tp_material int,
constraint CP_tb_material primary key (Codigo)) ;
create table tb_tipo_material (
codigo int not null,
Nome varchar(45),
constraint CP_tb_tipo_material primary key (codigo)) ;

create table tb_unidade_medida (
Codigo int not null,
Desricao varchar(8),
Preco decimal(11,2),
constraint CP_tb_unidade_medida primary key (Codigo)) ;

alter table tb_compra
add constraint fk_tb_compra foreign key (Cod_material)
references tb_material (Codigo);

alter table tb_material
add constraint fk_tb_material foreign key (Cod_tp_material)
references tb_tipo_material (codigo);
```

Quadro 20 – *Script* de criação do banco de dados para Mysql

APÊNDICE B – *Script* de atualização do banco de dados para Mysql

```
/*
Script de atualização de versão do banco de dados para Mysql
Projeto: GR 1
Versão anterior: 1.0
Nova versão: 2.0
Data: 19/11/2012 09:12:40
*/

alter table tb_compra
drop constraint fk_tb_compra;

alter table tb_material
drop constraint fk_tb_material;

create table tb_unidade_medida (
Codigo int not null,
Desricao varchar(8),
Preco decimal(11,2),
constraint CP_tb_unidade_medida primary key (Codigo)) ;

alter table tb_tipo_material drop Unidad_medida;

alter table tb_compra add Data_compra date;

alter table tb_compra
add constraint fk_tb_compra foreign key (Cod_material)
references tb_material (Codigo);

alter table tb_material
add constraint fk_tb_material foreign key (Cod_tp_material)
references tb_tipo_material (codigo);
```

Quadro 21 – *Script* de atualização do banco de dados para Mysql

APÊNDICE C – Relatório de diferenças entre as versões

```
/*
Relatório de diferenças entre versões
Projeto: GR 1
Versão anterior: 1.0
Nova versão: 2.0
Data: 19/11/2012 09:12:40
*/

* Tabelas adicionadas:
tb_unidade_medida - Unidade de medida

* Tabelas Alteradas:

Colunas adicionadas:
tb_compra.Data_compra - Data - permite nulo

Colunas excluídas:
tb_tipo_material.Unidad_medida - Texto Variável(8)
```

Quadro 21 – Relatório de diferença entre versões

ANEXO D – Casos de uso do sistema WebModeler 2.0

UC012 - Imprimir diagramas	
Descrição	Permite imprimir o diagrama gerado pelo sistema.
Atores	Analista de sistemas
Cenário Principal	1. O analista seleciona o projeto ao qual pertence o diagrama; 2. O analista seleciona a opção “Diagramas” no *menu* lateral do sistema; 3. O sistema abre uma tela solicitando a seleção do diagrama; 4. O analista seleciona a opção de modelar o diagrama; 5. O analista clica no ícone de impressão no *menu* de ferramentas; 6. O sistema abre uma janela solicitando a seleção da impressora; 7. O analista seleciona uma impressora e clica no botão “imprimir”; 8. O sistema imprime o diagrama.
Exceções	No passo 7: erro ao enviar o arquivo para impressora. O sistema não conseguiu enviar o arquivo para impressora, devido a algum problema com a impressora. O sistema volta a tela de modelagem de diagrama.
Pré-condições	Projeto, versão, tabelas e diagrama previamente cadastrados no sistema.
Pós-condições	Diagrama impresso.

Quadro 22 – UC012 – imprimir diagramas

UC013 - Importar estrutura de banco	
Descrição	Permite importar a estrutura de um banco de dados.
Atores	Analista de sistemas
Cenário Principal	1. O analista seleciona o projeto ao qual quer importar a estrutura; 2. O analista seleciona a opção “Importar” no *menu* lateral do sistema; 3. O sistema abre uma tela solicitando a *string* de conexão para o banco de dados; 4. O analista digita o usuário, senha e string de conexão e clica na opção “Conectar”; 5. O sistema importa as tabelas com suas colunas e seus relacionamentos, juntamente com um diagrama inicial do banco de dados.
Exceções	No passo 5: erro na conexão com banco de dados. O sistema não conseguiu conexão com o banco de dados informado. O sistema volta ao menu principal.
Pré-condições	Projeto e versão previamente cadastrados no sistema. O banco de dados informado necessita estar disponível.
Pós-condições	Tabelas, colunas e diagrama importados no sistema.

Quadro 23 – UC013 – importar estrutura de banco

UC014 - Cadastrar índices	
Descrição	Permite o cadastro de índices para as tabelas do SGBD.
Atores	Analista de sistemas
Cenário Principal	1. O analista seleciona o projeto ao qual pertence as tabelas que que incluir os índices; 2. O analista seleciona a opção “Tabelas” no *menu* lateral do sistema; 3. O sistema abre uma tela solicitando a seleção da tabela; 4. O analista seleciona a tabela; 5. O analista clica no botão “índices” na tela de edição de tabelas; 6. O sistema abre uma janela para o cadastro de índices; 7. O analista preenche as informações referentes ao índice e clica no botão “Gravar”; 8. O grava o índice para a tabela.
Exceções	No passo 7: erro, índice já existe. O sistema não consegue gravar o índice pois o mesmo já está cadastrado no sistema. O sistema volta a tela de edição de tabelas.
Pré-condições	Projeto, versão e tabelas previamente cadastrados no sistema.
Pós-condições	Índice Cadastrado.

Quadro 24 – UC014 – cadastrar índices

UC015 - Modelar diagramas UML	
Descrição	Permite a geração de diagramas UML.
Atores	Analista de sistemas
Cenário Principal	1. O analista seleciona o projeto ao qual quer importar a estrutura; 2. O analista seleciona a opção “Diagramas UML” no *menu* lateral do sistema; 3. O sistema abre uma tela solicitando a inclusão ou edição de diagramas; 4. O analista clica na opção “Criar diagrama”; 5. O analista preenche as informações referentes ao índice e clica no botão “Gravar”; 6. O grava o índice para a tabela.
Exceções	No passo 7: erro, índice já existe. O sistema não consegue gravar o índice pois o mesmo já está cadastrado no sistema. O sistema volta a tela de edição de tabelas.
Pré-condições	Projeto, versão e tabelas previamente cadastrados no sistema.
Pós-condições	Índice Cadastrado.

Quadro 25 – UC015 – modelar diagramas UML