CENTRO UNIVERSITÁRIO “EURÍPIDES DE MARÍLIA” - UNIVEM
TRABALHO DE CONCLUSÃO DE CURSO

OSCAR BRANCO DENIS

DESENVOLVIMENTO BASEADO EM MODELOS: DA TEORIA À PRÁTICA

MARÍLIA 2007

OSCAR BRANCO DENIS

DESENVOLVIMENTO BASEADO EM MODELOS: DA TEORIA À PRÁTICA

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação do Centro Universitário Eurípides de Marília, mantido pela Fundação de Ensino Eurípides Soares da Rocha, como requisito parcial para a obtenção do Título de Bacharel em Ciência da Computação.

Orientadora:
Profª. Drª. Maria Istela Cagnin Machado

MARÍLIA 2007

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Oscar Branco Denis

DESENVOLVIMENTO BASEADO EM MODELOS: DA TEORIA À PRÁTICA

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 7,0 (sete)

Orientador: Maria Istela Cagnin Machado

1°. Examinador: Kalinka R. L. J. Castelo Branco

2°. Examinador: Edmundo Sérgio Spoto

Marília, 19 de novembro de 2007.

Aos meus pais, que tanto

me apoiaram e amo.

AGRADECIMENTOS

Agradeço à de Deus por ter me concebido o dom da vida, e ser saudável podendo desenvolver as tarefas a mim designadas.

Aos meus pais, Oscar e Tânia por todo amor, carinho, apoio e muito sacrifício para que eu pudesse ter a melhor formação possível.

Aos meus irmãos, Emanuel e Eloá que tanto amo, e me ajudaram no que foi possível.

À minha noiva Luciane pelo amor, carinho e por ter me encorajado diversas vezes a retomar o trabalho.

À Profª. Drª. Maria Istela Cagnin Machado que desde o primeiro momento me orientou com muita dedicação, vontade, respeito, paciência, e pelo aprendizado que me proporcionou.

À todos os professores que durantes estes anos nesta instituição ajudaram na minha formação.

Aos meus amigos que me proporcionaram momentos felizes

À todos os funcionários desta instituição.

Enfim, a todos que me ajudaram a concluir esta etapa da minha vida.

DENIS, Oscar Branco. **DESENVOLVIMENTO BASEADO EM MODELOS: DA TEORIA À PRÁTICA.** 2007. 68f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2007.

RESUMO

A Engenharia de Software tem como objetivo produzir métodos e técnicas para o desenvolvimento de software e, conseqüentemente, produzir software com qualidade, rapidez e baixo custo. Nesse contexto, surgiu o Desenvolvimento Baseado em Modelos (*Model Driven Development* - MDD), que possibilita a geração do código fonte do sistema a partir de transformações de modelos em nível de abstração mais alto do que o código. Dentre as abordagens que apóiam esse tipo de desenvolvimento tem-se a MDA (*Model Driven Architecture*), definida pela OMG. Este trabalho tem como objetivo apresentar os principais conceitos envolvidos no MDD e o desenvolvimento de um sistema de pequeno porte com o apoio de uma ferramenta baseada na MDA, a fim de verificar a importância e a contribuição dessa nova forma de desenvolvimento na produção de software.

Palavras-Chave: MDD (Desenvolvimento Baseado em Modelos), MDA (Arquitetura Baseada em Modelos), Geração de código fonte.

DENIS, Oscar Branco. **DESENVOLVIMENTO BASEADO EM MODELOS: DA TEORIA À PRÁTICA.** 2007. 68f. Trabalho de Conclusão de Curso (Bacharelado em Ciência da Computação) – Centro Universitário Eurípides de Marília, Fundação de Ensino Eurípides Soares da Rocha, Marília, 2007.

ABSTRACT

The software engineering has as objective to produce methods and techniques for software development and, as a consequence, to produce software with quality, fastness and low price. In this context, emerged the Model Driven Development – MDD, which allows the generation of system source code from transformation of models in an abstraction level higher than the code. Among the approaches which support this kind of development we have the Model Driven Architecture, defined by OMG. This work has as objective to present the main concepts on the MDD and development of a low-shape system with the support of a tool based on the MDA, so we can verify the importance and the contribution of this new way of software production.

Keywords: MDD (*Model Driven Development*), MDA (*Model Driven Architecture*), Source code generation.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API: Application Programming Interface – Interface de Programação de Aplicativos

CASE: Computer Aided Software Engineering – Engenharia de Software Apoiada por Computador

CIM: Computation Independent Model – Modelo Independente de Computação

CWM: Common Warehouse Metamodel – Modelo de Repositório de Dados

MDA: Model Driven Architecture – Arquitetura Baseada em Modelos

MDD: Model Driven Development – Desenvolvimento Baseado em Modelo

MDSD: Model-Driven Software Development **–** Desenvolvimento de Software Dirigido por Modelo.

MOF: Meta Object Facility – Facilidade de Objetos Meta

MVC: Model View Control – Modelo Visão Controlador

ODBC: Open Data Base Connectivity – Conectividade Aberta de Base de Dados

OMG: Object Management Group – Grupo de Gerenciamento de Objetos

PIM: Platform Independent Model – Modelo Independente de Plataforma

PSM: Platform Specific Model – Modelo Específico de Plataforma

RAPDIS: Rules and Process for Development of Information Systems – Regras e Processos para Desenvolvimento de Sistemas de Informação.

SGDB: Data Base System Management - Sistema Gerenciador de Banco de Dados

TI: Technology Information – Tecnologia de Informação

UML: Unified Modelling Language - Linguagem de Modelagem Unificada

XMI: XML MetaData Interchange – Mudança Interna de Dados Meta

SUMÁRIO

INTRODUÇÃO

CONTEXTO

A Engenharia de Software tem como objetivo melhorar cada vez mais as várias tecnologias e ferramentas de elaboração de *software*, com o intuito de diminuir os custos, prazos e aumentar a qualidade do *software* (FERREIRA, 2005).

Uma forma de permitir isso é por meio do Desenvolvimento Baseado em Modelo – MDD, que centra as atenções diretamente na modelagem do "problema", sendo de grande importância para a construção do *software* (BELIX, 2006).

Existem várias abordagens, técnicas, métodos e ferramentas que apóiam o MDD. Dentre as abordagens mais utilizadas tem-se a MDA (OMG, 2003). Essa abordagem tem como objetivo o desenvolvimento de um modelo inicial, livre de considerações da plataforma onde o sistema será implantado. Este modelo inicial é então transformado – segundo um conjunto de transformações - em um modelo que considera as características especificas da plataforma, este último subseqüentemente transformado em código fonte (OMG, 2003).

Com relação às ferramentas de apoio ao MDA existem várias. Dentre elas, algumas atendem todo o processo de desenvolvimento baseado na MDA e outras não, ou seja, algumas ferramentas apóiam completamente o processo e outras apenas de maneira parcial.

MOTIVAÇÃO E JUSTIFICATIVA

Com a crescente demanda de novas empresas e a informatização das já existentes, há um aumento significativo na quantidade de *softwares* que apóiam no gerenciamento de tais empresas.

As empresas necessitam cada vez mais de sistemas complexos para poderem controlar seus negócios. Com isso, a elaboração dos mesmos torna-se demorada e, conseqüentemente, custosa, pois é necessário despender tempo e esforço significativos dos desenvolvedores para atingir os objetivos pretendidos.

Na maioria das vezes os *softwares* não atendem todos os requisitos previamente estabelecidos, seja por falta de testes ou por levantamento de requisitos efetuado incorretamente. Isso faz com que há um atraso na entrega da versão definitiva do *software*.

Para que a elaboração de um *software* seja bem sucedida, é necessário que as fases de levantamento de requisitos e de análise sejam executadas cuidadosamente. A fase de análise é considerada mais importante do que a fase de codificação, ou seja, quanto mais detalhada for a análise menor é a ocorrência de erros durante a fase de codificação.

Com o desenvolvimento baseado em modelos, por exemplo, utilizando a arquitetura baseada em modelos (MDA), os desenvolvedores têm a possibilidade de se concentrarem e despenderem mais tempo durante as fases de análise e projeto, pois o projeto é automaticamente transformado no código fonte por meio de ferramentas computacionais específicas para isso. Assim, há redução de gastos e aumento na agilidade da entrega do produto final sem ultrapassar o cronograma estipulado.

OBJETIVOS

O objetivo do trabalho é estudar a arquitetura baseada em modelos e efetuar um estudo de caso utilizando uma ferramenta de apoio a fim de observar as vantagens obtidas com a utilização dessa nova forma de desenvolvimento de *software*.

ORGANIZAÇÃO

Este trabalho está organizado em quatro capítulos. No Capítulo 2 são abordados os principais conceitos da arquitetura baseada em modelos – MDA, envolvendo seus modelos com suas transformações. Ainda no Capítulo 2 são apresentadas as várias tecnologias existentes e quais serão utilizadas para o desenvolvimento deste trabalho. Ainda neste capítulo são apresentadas algumas características de uma ferramenta de apoio ao MDA, denominada RAPDIS. No Capítulo 3 é apresentado um estudo de caso de desenvolvimento baseado em modelo com o apoio da ferramenta RAPDIS, discutindo os problemas encontrados e as

soluções tomadas. No Capítulo 4 é apresentada a conclusão do trabalho, ressaltando as suas limitações e fornecendo sugestões de trabalhos futuros.

2. DESENVOLVIMENTO BASEADO EM MODELOS (MDD)

O MDD (*Model Driven Development*) – Desenvolvimento Baseada em Modelos considera que no desenvolvimento de *software* a modelagem do problema é muito mais importante do que o desenvolvimento do código fonte. Com isto, reduz-se a importância do código e as atenções ficam mais concentradas no que realmente importa: construir uma aplicação final que funcione de acordo com os requisitos especificados e fornecidos pelo cliente e usuários (COSTA *et al.*, 2006). Nesse tipo de desenvolvimento, a construção do código é feita por meio de ferramentas específicas que realizam a geração automática da estrutura e de partes do código, facilitando o desenvolvimento do *software* (COSTA, et al , 2006).

MDD pode ser apoiado por mais de uma abordagem (BELIX, 2006), uma vez que a OMG (*Objetc Management Group*) definiu um conjunto de padronizações que possibilita a definição de abordagens de desenvolvimento baseadas em modelos independentes de plataforma, como é o caso do MDA (*Model Driven Architecture*), que tem como foco principal a separação entre a especificação e a implementação, em uma plataforma específica (OMG (2003b) **apud** BELIX (2006)). Outras abordagens de desenvolvimento baseada em modelos são encontradas na literatura, como é o caso de MDSD (*Model-Driven Software Development*) e *Software Factories* da Microsoft (BELIX, 2006)

Sucintamente, a abordagem MDSD tem como objetivo a modelagem e a metamodelagem e não está vinculada a nenhum metamodelo específico (como o MOF (*Meta Object facylit*)) e a nenhuma linguagem de modelagem específica (como a UML – *Unified Modeling Language*) (BELIX, 2006). A abordagem *Software Factories* trata de modelos simples, porém formais e altamente focados em aspectos específicos de requisitos arquitetura e implementação (BELIX, 2006).

Dentre as abordagens existentes, a MDA é de interesse deste trabalho, pois é um padrão da OMG, que é mundialmente reconhecido. Tal abordagem está descrita detalhadamente na Seção 2.1.

2.1 Arquitetura Orientada por Modelos (MDA)

A MDA (*Model Driven Architecture)* - Arquitetura Orientada por Modelos - é uma tecnologia que está sendo incorporada por departamentos de TI (Tecnologia da Informação) de grandes corporações. Corresponde ao que se chama de linguagem de quinta geração, por envolver conceitos formalizados para a geração do código fonte e por se tratar de um novo paradigma que engloba novas tecnologias para a construção do *software* (KRIESER, 2006).

MDA é utilizada para que possa ser feita separação da especificação e da implementação de um sistema em uma plataforma. O MDA permite especificar um sistema independente de plataforma, transformar a especificação do sistema em uma determinada plataforma e permite definir uma plataforma especifica para fazer a implementação do sistema (BELIX, 2006).

Esta abordagem foi definida em 2000 pela OMG com o objetivo de solucionar o problema das mudanças de tecnologia e do negócio nas aplicações (SANTOS *et al.*, 2005).

A MDA auxilia na correção dos problemas relacionados à modelagem, à sincronização dos modelos com o código fonte desenvolvido, à ambigüidade desses modelos entre outros problemas que ocorrem na elaboração do *software* (MAIA, 2006).

A linguagem de modelagem utilizada pela MDA é a UML *(Unified Modeling Language)* - Linguagem de Modelagem Unificada (Versão 2.0), que é uma linguagem de diagramação, visualização e documentação de modelos de sistema de *software* orientados a objetos (KRIESER, 2006).

Utilizando a MDA no desenvolvimento de *software* podem-se obter diversos benefícios como (BELIX, 2006):

- **Redução de custos:** a geração de boa parte do código fonte faz com que menos horas sejam despendidas durante a fase de codificação.
- **Redução de prazos:** com o aumento da produtividade diminuem-se os prazos.
- **Interoperabilidade:** Vários PSMs criados a partir de um mesmo PIM podem conter ligações entre eles, os quais são conhecido na MDA pelo rótulo de pontes*(Bridges).* Uma *ponte* pode ser construída através das especificações técnicas das plataformas referentes aos modelos PSMs e de como os elementos existentes no modelo PIM foram transformados nos modelo PSMs.(BELIX *et al*.,2006).

- **Portabilidade:** a reutilização de componentes e modelos permite a redução da complexidade dos sistemas.

2.2 Conceitos

Um dos conceitos da MDA está na idéia de utilizar a linguagem de modelagem como linguagem de programação (BELIX, 2006). Para isso, no desenvolvimento de um *software* a modelagem representa o projeto, ou seja, como o *software* tem que ser feito, para diminuir a complexidade durante a elaboração do mesmo.

Antes da MDA, os modelos eram construídos para facilitar a comunicação entre o desenvolvedor e o projetista, responsável pelo projeto a ser desenvolvido. Com a MDA, os modelos necessitam de uma maior precisão para que possam fazer parte da produção do *software* (MAIA, 2006).

2.3 Modelos Utilizados

Os modelos utilizados na MDA são considerados modelos Abstratos e modelos Concretos.

Os modelos Abstratos são apenas para classificação da Taxonomia, ou seja, para o estudo do caso a ser desenvolvido (MAIA, 2006).

Por outro lado, os modelos Concretos são constituídos no processo de desenvolvimento, são considerados todos como modelos de sistemas com exceção do Modelo de Negócio. Cada um dos modelos representa de um ponto de vista diferente o sistema, ou seja, cada modelo mostra uma visão diferente de cada parte do sistema (MAIA, 2006). O Modelo de requisito, representado na MDA com o CIM *(Computation Independent Model)* é a representação mais abstrata do sistema, enquanto que o PIM *(Platform Independent Model)* é o menos abstrato, pois apresenta algumas considerações técnicas e a parte lógica do sistema (MAIA, 2006).

O PIM é um refinamento do modelo de Requisito e o PSM é um refinamento de um PIM e é descrito em termos específicos para uma plataforma específica onde o sistema é

implementado (MAIA, 2006). Os Modelos Físicos representa artefatos físicos utilizados para execução do sistema, como por exemplo, arquivos e nós computacionais (MAIA, 2006).

Os modelos utilizados na MDA são basicamente divididos em três modelos diferentes: o CIM *(Computation Independent Model)* – Modelo Computacional Independente que também é conhecido como Modelo de Requisitos, o PIM *(Platform Independent Model)* – Modelo Independente de Plataforma e o PSM *(Platform Specific Model)* – Modelo para Plataforma Específica. Cada um desses modelos será detalhado nas subseções a seguir.

2.3.1 Modelo Independente Computacional – CIM

O CIM oferece uma visão do sistema por meio da perspectiva independente dos detalhes computacionais (MAIA, 2006).

Este modelo representa os requisitos do sistema, que são implementados por especialistas. Ele é voltado para o especialista do domínio e não representa a realização de funcionalidades do sistema (SANTOS, et al, 2005).

O CIM, em termos MDA, pode ser considerado o modelo do Ponto de Vista da Empresa, que tem a perspectiva de modelo de negócio e é extremamente útil para comunicar as necessidades do cliente aos arquitetos.

Um sistema de *software* pode suportar algumas partes do CIM, mas o CIM fica independente do *software*. O CIM não é derivado automaticamente para o PIM, porque a escolha das partes para serem suportadas pelo sistema é feita por humanos (BELIX, 2006), com ilustra Figura 1.

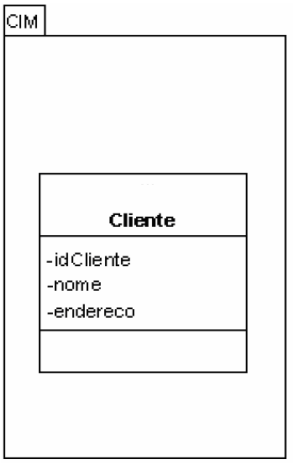

FIGURA 1 – Modelo Independente Computacional – CIM

2.3.2 Modelo Independente de Plataforma – PIM

Um projeto MDA é iniciado com um CIM que dá origem a um PIM, que de preferência deve ser projetado em UML por meio de diagramas.

Este modelo representa o sistema de um ponto de vista independente da plataforma. O objetivo é que ele mude apenas quando alguma lógica de negócio sofrer mudanças (SANTOS *et al*., 2005).

O PIM, como o próprio nome diz, indica um modelo independente de plataforma e tecnologia, apresenta um alto nível de abstração. Um modelo PIM descreve um modelo de negócio e os detalhes de como as funcionalidades do sistema serão implementadas sem especificar detalhes sobre as tecnologias utilizadas (FERREIRA, 2005), como apresentado na Figura 2.

Este modelo apresenta as regras de negócios e as funcionalidades menos distorcidas possível pela tecnologia. Neste modelo podem ser tratados conceitos como persistência, nível de segurança e suporte transacional, fazendo com que a elaboração do PSM seja mais precisa (MAIA, 2006).

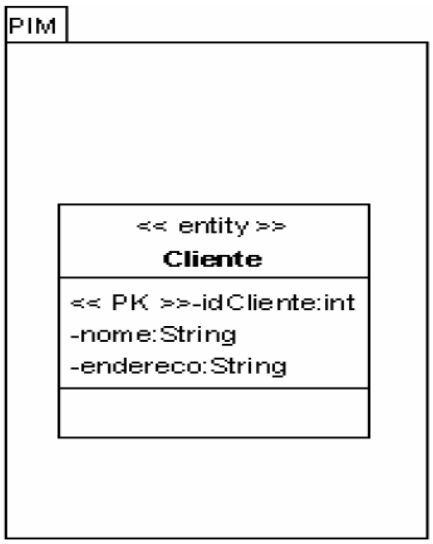

FIGURA 2 – Modelo Independente de Plataforma – PIM (MAIA, 2006)

2.3.3 Modelo Específico de Plataforma – PSM

Como o próprio nome indica PSM é um Modelo Específico de Plataforma. Ele agrega as especificações do PIM com os detalhes que determinam como o sistema usa um tipo particular de plataforma (SANTOS, et al, 2005), como ilustrado na Figura 3.

Como a implementação de um sistema pode cobrir diferentes plataformas (banco de dados, camada de apresentação *web*, servidor de aplicações, etc), diferentes PSM's podem ser gerados a partir de um único PIM (BELIX, 2006).

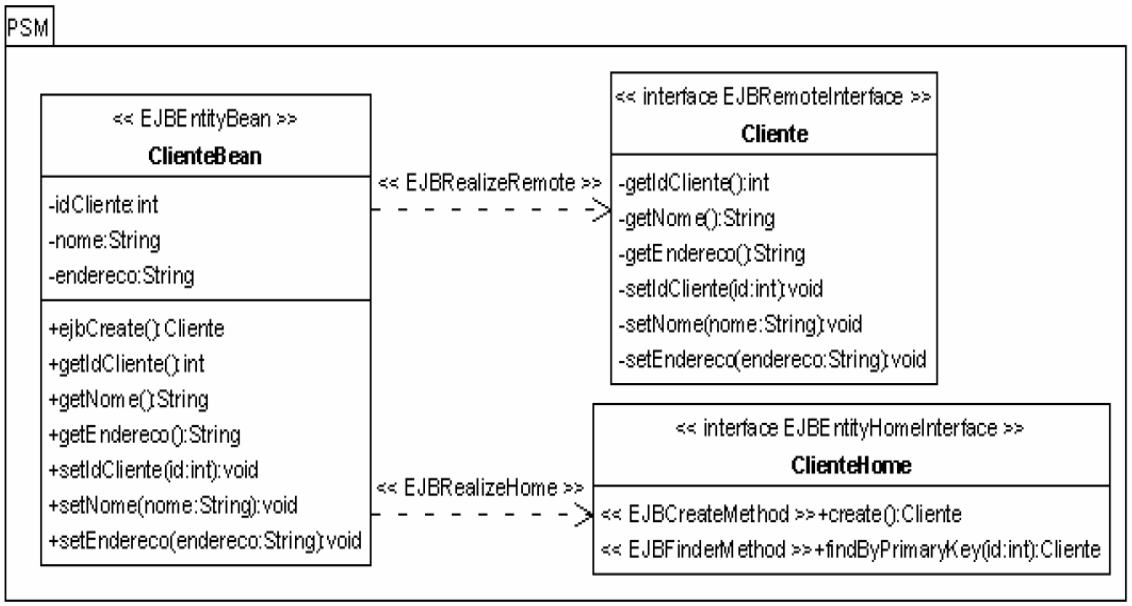

FIGURA 3 – Modelo Específico de Plataforma – PSM (MAIA, 2006)

2.3.4 Código – Fonte

A geração do código fonte é dada através da transformação do PSM em código. Para que a geração deste código possa ser o mais exata possível, o PSM tem que fornecer mais ou menos detalhes para a criação do código, dependendo da sua finalidade (BELIX, 2006).

A criação do código fonte tem como objetivo transformar um modelo abstrato de alto nível em um programa (código fonte), a fim de facilitar a elaboração do *software*. A transformação do Modelo de Sistema de Informação é basicamente a tradução do que foi especificado por meio da UML, para um código fonte na linguagem específica (MORGADO, 2007).

O PSM, para ser utilizado na geração do código com sucesso, tem que conter todas as informações necessárias para construir e operar o sistema (BELIX, 2006).

2.3.5 Transformação Entre os Modelos Utilizados

As transformações são feitas de um modelo em um nível mais alto para um modelo em um nível mais baixo, fazendo com que, em um primeiro momento, o modelo do nível mais alto seja a entrada de dados e, após a transformação, o modelo em um nível mais baixo seja a saída de dados. A transformação básica modelo–modelo é a transformação PIM-PSM (MAIA, 2006), como apresentado na Figura 4.

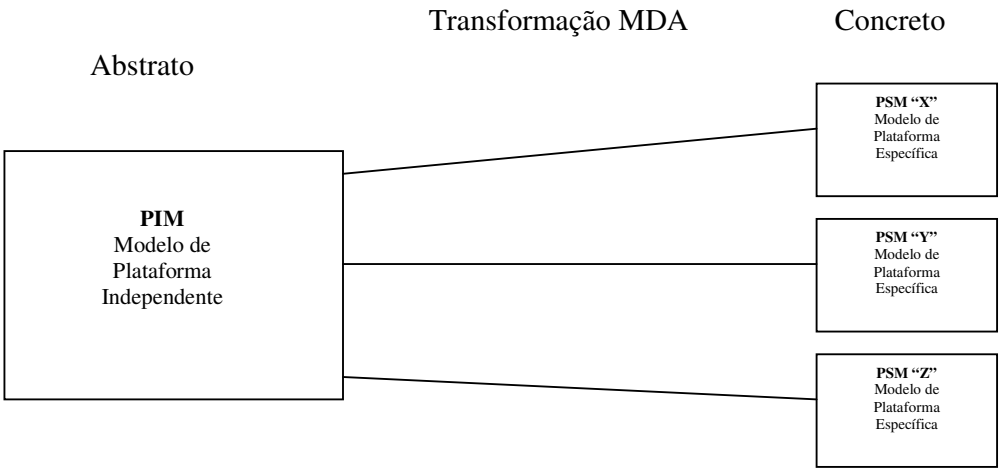

FIGURA 4 - PIM em PSMs (MAIA, 2006).

O MDA está dividido em quatro métodos de transformação (MAIA, 2006). Esses métodos são explicados a seguir.

- Transformação manual – é definida como a elaboração do projeto, ou seja, a aquisição de informação para criar um projeto que proporcione uma implementação em conformidade com os requisitos previamente definidos. Após todos os requisitos definidos, a contribuição de MDA vem com a distinção explícita do PIM (MAIA, 2006), ou seja, é feita a transformação do CIM em PIM.

- Transformação de PIM preparado com perfil – um especialista manipula o PIM, que realiza marcações utilizando um perfil UML é independente de plataforma (BELIX, 2006). Transformando o PIM em um PSM expresso por um segundo perfil UML, este perfil UML é específico de plataforma (MAIA, 2006).
- Transformação utilizando padrões e marcações – são utilizados para especificar os elementos no PIM, seguindo um padrão determinado, que serão transformados em instâncias de um novo padrão, o PSM (MAIA, 2006).
- Transformação automática – quando o PIM é totalmente completo, ou seja, com todas as informações necessárias para a implementação. O desenvolvedor pode especificar diretamente no modelo, utilizando uma linguagem de ações. Essa linguagem faz com que o PIM seja computacionalmente completo e automaticamente transformado no código fonte do programa. (MAIA, 2006).

Na Figura 5 são ilustradas as passagens das transformações do nível mais alto de abstração para o nível mais baixo de abstração, ou seja, desde o CIM até a geração do código fonte.

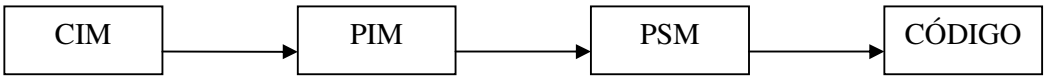

FIGURA 5 – Modelos e Transformações MDA (MORGADO *et al*., 2007)

2.3.6 Tecnologias de Apoio

Para o desenvolvimento dos modelos são utilizadas diversas tecnologias de apoio como: UML que é considerada a linguagem para elaborar a representação visual orientada a objetos do *software*, MOF (*Meta Object Facility*) (BELIX, 2006) que define uma linguagem abstrata para a descrição dos modelos, XMI (*XML MetaData Interchange*) (FERREIRA,2005) que define um conjunto de regras que mapeiam os meta-modelos no MOF e os modelos em

documentos XML (*eXtensible Marckup Language*) (MAIA, 2006). Cada uma dessas tecnologias está apresentada mais detalhadamente a seguir.

- **UML**

A *Unified Modeling Language* firmou-se como a linguagem de modelagem dominante na comunidade de desenvolvimento de *software*. Sua popularidade é um dos fatores fundamentais para o seu sucesso no MDA e por ser uma linguagem que está em constante evolução (BELIX, 2006).

Por ser considerada uma linguagem de modelagem padrão no desenvolvimento orientado a objetos, a UML apóia a criação de modelos independentes de plataforma (MAIA, 2006). Além disso, por oferecer uma abordagem multi-visão, a UML fornece aos desenvolvedores diversas vantagens com a divisão das áreas de interesse durante o processo de modelagem de um *software.* São oferecidos diferentes diagramas para vários propósitos durante o processo de desenvolvimento (BELIX, 2006):

- Diagrama de Caso de Uso;
- Diagrama de Classes;
- Diagrama de Pacotes;
- Diagrama de Seqüência de Evento;
- Diagrama de Colaboração;
- Diagrama de Estados;
- Diagrama de Componentes;
- Diagrama de Distribuição.

Dos diagramas citados, será feito um rápido comentário dos mais utilizados para a construção dos casos dentro da MDA.

O diagrama de casos de uso é utilizado para ilustrar os casos de usos com seus autores e relações. Sua utilização no desenvolvimento visa a representação do caso em análise, descriminando com detalhes os autores e os requisitos funcionais (MORGADO, 2007).

O diagrama de classes é o mais utilizado para o desenvolvimento de sistemas OO baseado em modelos, uma vez que é composto por classes, interfaces e seus relacionamentos. As classes contêm métodos e atributos (BELIX, 2006).

Os diagramas de classes descrevem as classes que formam a estrutura do sistema e suas relações. Além de exibir os tipos dos atributos, a visibilidade, a multiplicidade das relações e diversas restrições. Ao final do processo de modelagem, pode ser traduzido em uma estrutura de código que servirá de base para a implementação do código (DEBONI, 1998).

- **CWM**

O objetivo do *Common Warehouse Metamodel* (CWM) é modelar o domínio e disponibilizar a fácil troca de meta-dados de ferramentas *datawerehouse* (MAIA, 2006).

CWM organiza a estrutura de mapeamento entre os elementos de entrada e os de saída por meio de uma especificação, fazendo com que esta especificação promova um apoio à definição de transformações de modelos de dados baseados em meta-modelos que tratam as transformações (MAIA, 2006).

- **MOF**

Na MDA o *Meta Object Facility* tem um papel fundamental, pois é ele que permite que o mapeamento das transformações entre modelos de diferentes meta-modelos seja definido nos termos de construção (MAIA, 2006), pois o modelo e o meta-modelo são descrições de um sistema ou parte dele (BELIX, 2006).

O MOF é um padrão criado pela OMG para a definição dos meta-modelos de seus padrões, como a especificação CWM, XMI e a própria UML. A OMG utiliza uma arquitetura dividida em quatro camadas de modelo: M0, M1, M2 e M3 (BELIX *et al*., 2006), como apresentado na Figura 6.

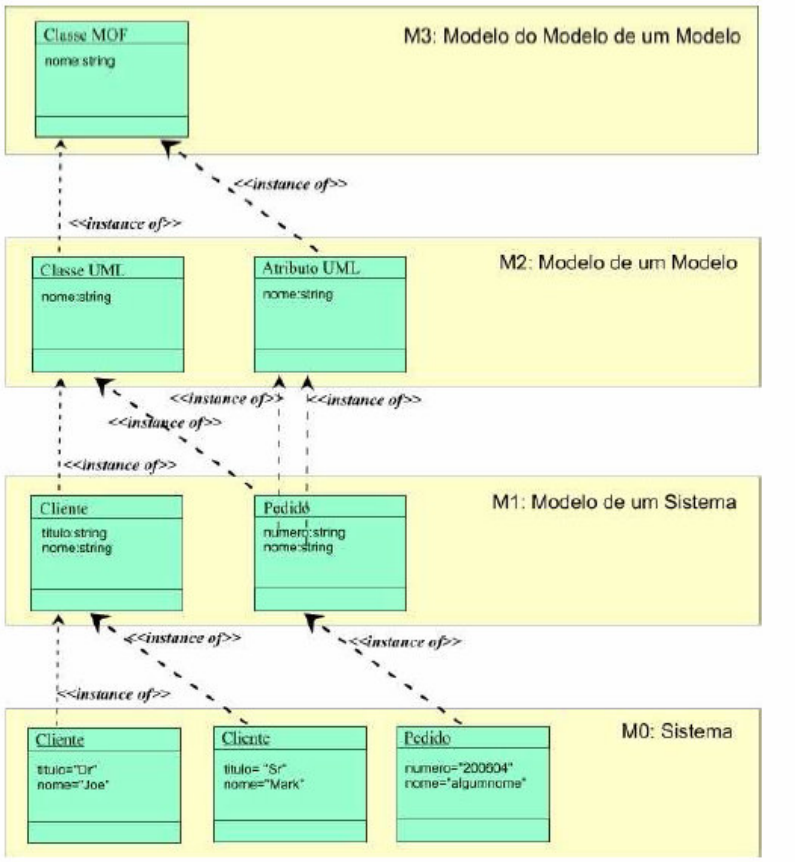

FIGURA 6 – Camadas definidas pela OMG (MAIA, 2006)

MOF não é uma gramática, é uma linguagem utilizada para descrever uma estrutura de objetos, ou seja, por meio do MOF é possível especificar uma linguagem formalmente (MAIA, 2006).

- **XMI**

O padrão XMI permite um intercâmbio de metadados entre ferramentas de modelagem, baseadas em UML, ou seja, define um conjunto de regras que interligam os metamodelos baseados em MOF (MAIA, 2006).

Por meio do padrão XMI são efetuados o armazenamento e o intercâmbio de modelos UML na forma de documentos XML. As principais características do padrão XMI são (FERREIRA, 2005):

- Possibilita que os objetos descritos sejam representados como padrão no XML, permitindo o intercâmbio dos objetos;
- Especifica, a partir de modelos, como criar XML;
- Especifica a criação de documentos XML básico, que após a adição de elementos a estrutura possa evoluir;
- Os usuários não necessitam conhecer integralmente a linguagem XML para utilizar o padrão.

O XMI é um arquivo texto que pode ser alterado por qualquer linguagem que manipule arquivos texto e edite as transformações UML para UML ou UML para código-fonte (BELIX, 2006).

2.4 RAPDIS: Uma ferramenta de apoio a MDA

2.4.1. Visão Geral

São inúmeras as ferramentas de apoio disponíveis para a criação de *software* baseada na arquitetura orientada por modelos. Para obter a indicação de algumas destas ferramentas, basta fazer uma rápida procura na Internet.

Algumas destas ferramentas podem se adquiridas por meio da compra, como é o caso da ferramenta da Borland (*Together Architect*[1]), da *Computware* (Optmal J[2]), da *Interactive Objects* (Arcstyler[3]), dentre outras, ou sem custo algum, como é o caso da ferramenta RAPDIS[4], desenvolvida pelo grupo de Gestão Estratégica de Tecnologia da Informação - GETI da Universidade Federal do Rio de Janeiro – UFRJ, da ferramenta da M1 Global (MDE Studio[5]) e da ferramenta AndroMDA[6].

Por se tratar de um trabalho de conclusão de curso será adotada uma ferramenta acadêmica e sem custo. A ferramenta que melhor se adequou às necessidades e aos objetivos deste trabalho foi a RAPDIS.

[1] http://www.bolrland.com
[2] http://www.computware.com.br
[3] http://www.arcstyler.com
[4] http://www.geti.dcc.ufrj.br/projetos/rapdis
[5] http://www.m1global.com
[6] http://www.andromda.org

2.4.2. Rules and Process for Development of Information Systems - RAPDIS

RAPDIS é um *software* gratuito para fins acadêmicos desenvolvido em *Object Pascal* para plataforma Windows. A Ferramenta RAPDIS é um ambiente CASE (*Computer Aided Software Engieering*) que possibilita a aplicação da MDA. Esse ambiente tem como objetivo projetar modelos e automatizar as possíveis transformações do projeto (MORGADO, 2007). Na Figura 7 é ilustrada a tela de abertura do RAPDIS.

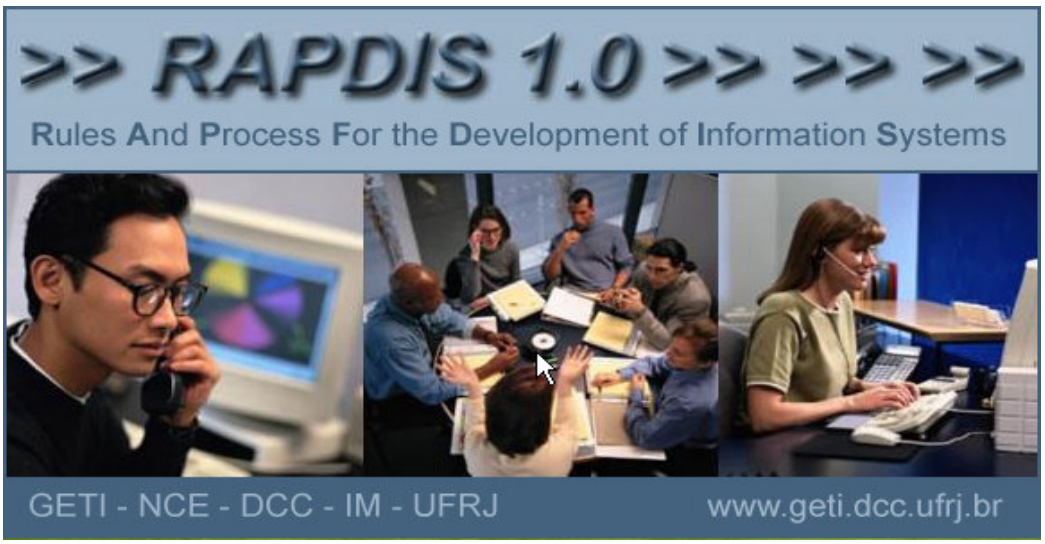

FIGURA 7 – Tela principal do RAPDIS

Como o documento da OMG que explica a MDA não definiu exatamente como construir o CIM, PIM e o PSM, e não informa como transformar um modelo de alto nível de abstração em um outro de baixo nível, o grupo GETI com a dificuldade de se aplicar a MDA baseou-se não apenas na definição da MDA, como também em métodos de modelagem e de transformações para efetuar o desenvolvimento da RAPDIS (MORGADO, 2007).

Além da ferramenta, é fornecido também um processo que consiste na execução dos seguintes procedimentos: “Construir o Modelo de Negócio”, “Construir o Modelo de Sistema de Informação” e “Programar o Sistema” (MORGADO, 2006).

O primeiro modelo a ser construído é o modelo de negócio que vai representar o CIM da MDA (MORGADO, 2006), como ilustrado na Figura 8 por meio dos diagramas de atividade e de casos de uso. O modelo de negócio é o responsável por definir Objetivos, Recursos, Termos, Regras de Negócio e Processo de Negócio para melhor definir o sistema de informação (MORGADO, 2007).

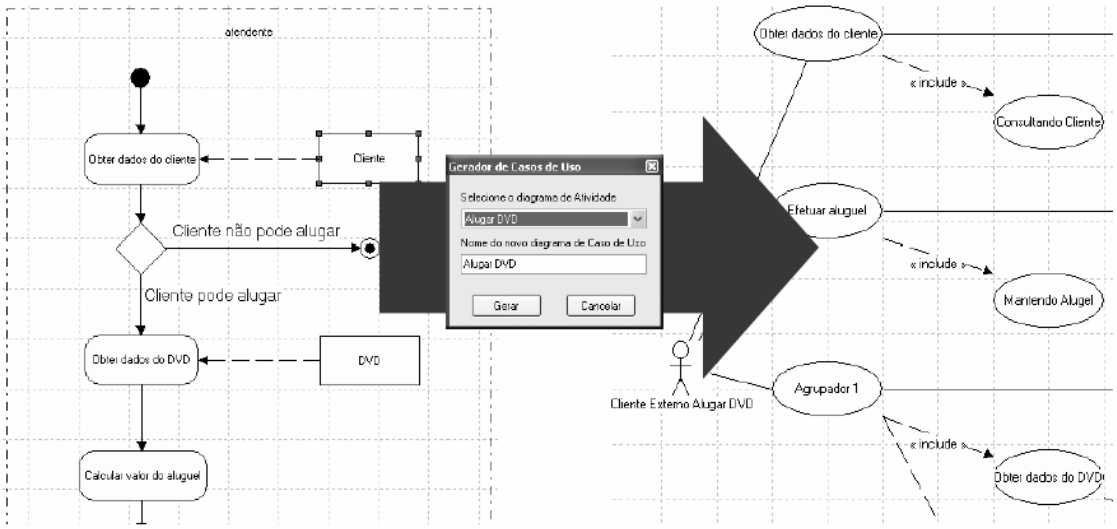

FIGURA 8 – Modelo CIM construído na RAPDIS (MORGADO, 2006).

A atividade "Construir o Modelo de Sistema de Informação" corresponde a construção do PIM (MORGADO, 2006), como apresentado na Figura 9 por meio do diagrama de classes.

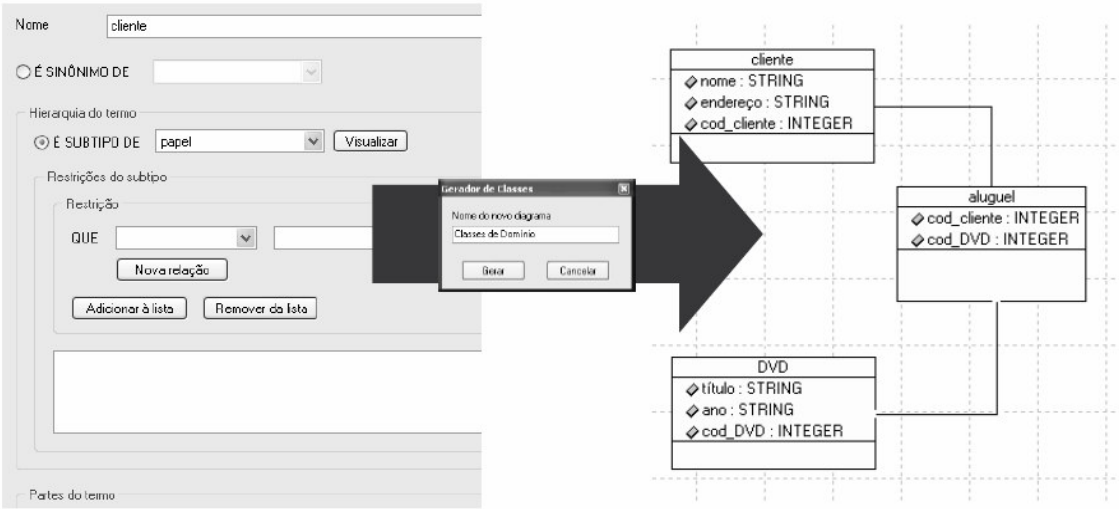

FIGURA 9 – Modelo PIM construído na RAPDIS (MORGADO, 2006)

A atividade "Implementar o Sistema" é responsável por fazer a transformação PIM para PSM e do PSM para o código (MORGADO, 2006), como ilustrado na Figura 10.

FIGURA 10 – Utilização do Gerador de Código (MORGADO, 2006)

Para facilitar o entendimento da MDA, bem como apresentar a aplicabilidade da ferramenta RAPDIS, são apresentados no Capítulo 3 os passos para a elaboração de um determinado projeto.

3 DESENVOLVIMENTO BASEADO EM MODELO APOIADO PELA FERRAMENTA RAPDIS

3.1 Elaboração do Projeto

Para analisar a aplicabilidade de ferramentas no desenvolvimento baseado em modelos, utilizou-se neste trabalho a ferramenta RAPDIS para apoiar o desenvolvimento de um sistema de pequeno porte de locação de carros. Esse sistema tem como objetivo efetuar o controle de aluguel de carros. O documento de requisitos completo desse sistema está apresentado no Anexo A.

A Ferramenta RAPDIS oferece funcionalidades para a criação do Modelo de Negócio e do Modelo de Sistemas de Informação. Neste trabalho foi utilizado apenas o Modelo de Sistemas de Informação que permite a especificação do sistema por meio dos diagramas de classes, de casos de uso e de seqüência.

Uma vez que a ferramenta utiliza essencialmente o diagrama de classes para a geração do código fonte do sistema, somente esse diagrama foi considerado neste trabalho. Os diagramas de casos de uso e de seqüência são utilizados pela ferramenta para criar os menus na tela principal do sistema e a interface gráfica dos cadastros por meio de caixas de texto. No entanto, observou-se que quando o sistema é gerado em Java, não aparecem botões para a manipulação dos dados (inserção, remoção, alteração e consulta) na interface dos cadastros, tornando-a inviável para o uso.

A criação da interface gráfica por meio de *grids*, que é gerada a partir do diagrama de classes, foi a utilizada para que o sistema funcionasse adequadamente, possibilitando que os testes pudessem ser realizados e, consequentemente, problemas fossem detectados.

Neste trabalho, o PIM foi elaborado pelo engenheiro de software com o apoio da ferramenta RAPDIS. A partir desse modelo, criou-se automaticamente o PSM, de acordo com a plataforma selecionada pelo programador, no caso, projeto Eclipse[7] em Java 5.0. A ferramenta RAPDIS oferece mais duas plataformas que podem ser selecionadas que são Delphi 7.0 e Java 5.0. Posteriormente, a partir do PSM, gerou-se o código fonte do sistema. Ressalta-se que o CIM não foi implementado na ferramenta, uma vez que não foi encontrado material que apoiasse a elaboração desse modelo.

[7] www.eclipse.org/

3.1.1 Elaboração do PIM

A elaboração do PIM, por meio do diagrama de classes utilizando notação UML, foi efetuada na própria Ferramenta RAPDIS, como ilustrado nas Figuras 11 e 12.

.

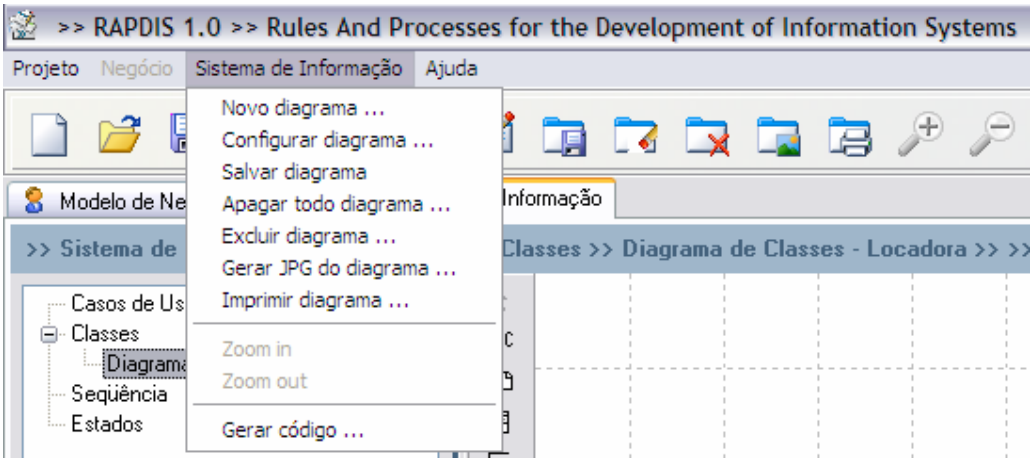

FIGURA 11 - Criando Novo Diagrama

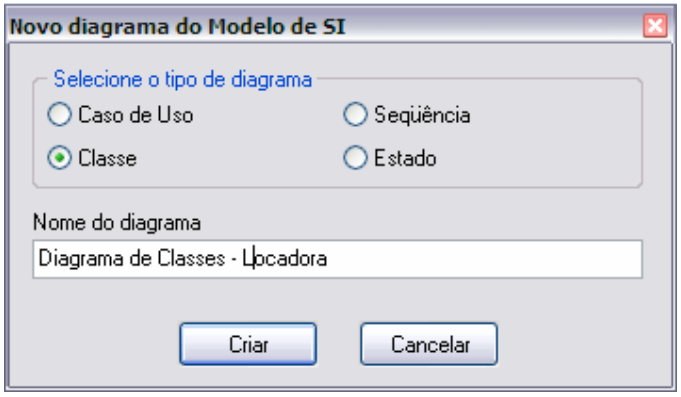

FIGURA 12 – Criando Diagrama de Classes (PIM)

Durante a criação das classes dos diagramas, o desenvolvedor deve classificá-las como Classe de Domínio, de Interface e de Controle a fim de aplicar o padrão de projeto MVC (*Model View Control*) (Gamma *et al.*, 1994). Neste projeto, as classes do sistema foram classificadas apenas como de domínio uma vez que optou-se por utilizar a interface gráfica padrão gerada pela ferramenta que é baseada em *grids*.

As classes criadas e a classificação definida para cada uma estão descritas na Tabela 1.

TABELA 1 – Dados para a Criação das Classes.

Nome da Classe	Tipo de Classe
Cliente	Domínio
Carro	Domínio
Pagamento	Domínio
Locação	Domínio

Com as classes já criadas é necessário determinar os atributos para cada uma a fim de que a ferramenta futuramente crie os campos das tabelas do *software*.

Os atributos criados para cada classe são os descritos nas Tabelas 2, 3, 4 e 5.

TABELA 2 - Atributos da Classe Cliente

Nome	Visibilidade	Tipo
Codigo	Privado	Integer
Nome	Privado	String

TABELA 3 - Atributos da Classe Carro

Nome	Visibilidade	Tipo
Placa	Privado	String
Marca	Privado	String
Modelo	Privado	String
Locado	Privado	String

TABELA 4 - Atributos da Classe Pagamento

Nome	Visibilidade	Tipo
Código	Privado	Integer
Descrição	Privado	String

TABELA 5 - Atributos da Classe Locação

Nome	Visibilidade	Tipo
Numero	Privado	Integer
Data	Privado	String
Data Devolução	Privado	String
Cliente	Privado	Integer
Carro	Privado	String
Pagamento	Privado	Integer

Posteriormente, foram especificados as multiplicidades e os nomes de cada relacionamento definido anteriormente, conforme apresentado nas Tabelas 6, 7, 8.

TABELA 6 - Relacionamento entre as classes Cliente e Locação

Papel da Associação	Papel na Classe (Cliente) Multiplicidade	Papel na Classe (Locação) Multiplicidade
Efetua	1..1	1..*

TABELA 7 - Relacionamento entre as classes Carro e Locação

Papel da Associação	Papel na Classe (Carro)	Papel na Classe (Locação)
Possui	0..*	1..1

TABELA 8 - Relacionamento entre as classes Pagamento e Locação

Papel da Associação	Papel na Classe(Pagamento)	Papel na Classe (Locação)
Tem	0..*	1..1

Na Figura 13 é apresentado o diagrama de classes contendo as classes, atributos, relacionamentos de associação entre as classes, e a multiplicidade de cada relacionamento. Ressalta-se que os métodos não foram especificados, pois a ferramenta RAPDIS gera

automaticamente o código fonte dos métodos de inserção, remoção e alteração de objetos. Caso haja necessidade de outros métodos, os mesmos devem ser especificados e implementados manualmente pelo programador.

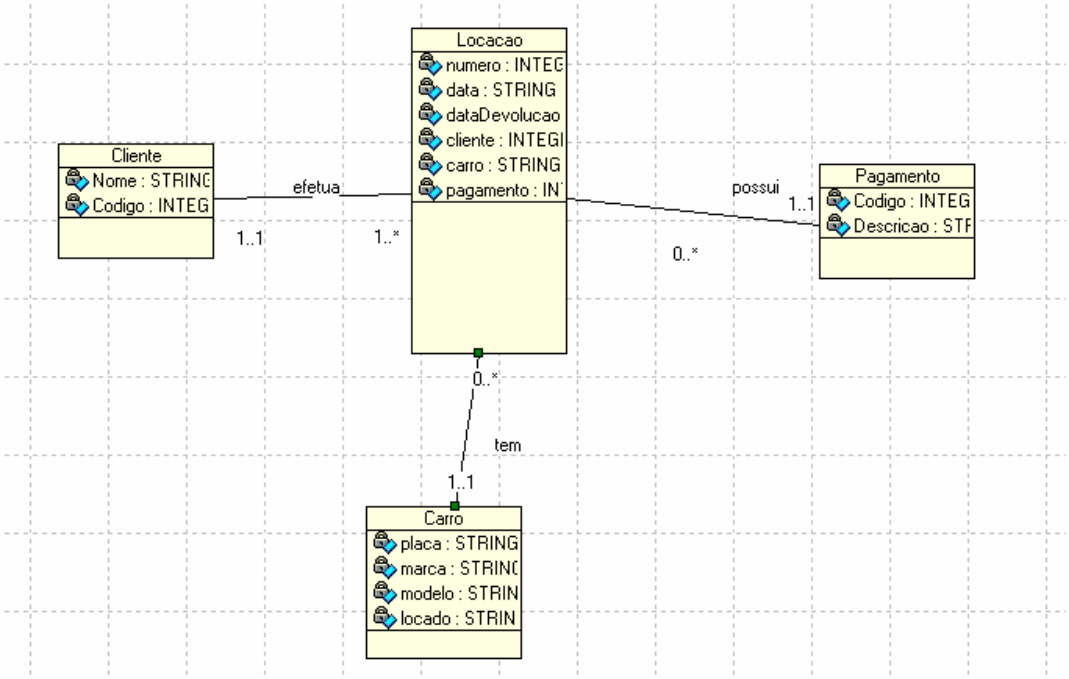

FIGURA 13 – Diagrama de classes completo do sistema de locação de carros

3.1.1.1 Geração do PSM e do Código Fonte

A geração do PSM (Modelo Específico de Plataforma) e do Código Fonte é feita automaticamente pela própria ferramenta, como comentado anteriormente. O PSM (Modelo Específico da Plataforma) como o próprio nome diz, representa as especificidades da plataforma alvo em que o código fonte é gerado.

A ferramenta oferece três opções para a criação deste modelo. A primeira opção é para a criação de um PSM para a plataforma Delphi 7.0, a segunda opção é para a criação do PSM para a plataforma Java 5 e a terceira opção é para a criação do PSM para a plataforma Java com o projeto no Eclipse, como ilustrado na Figura 14 Em todos os casos, o banco de dados do sistema é gerado no Paradox.

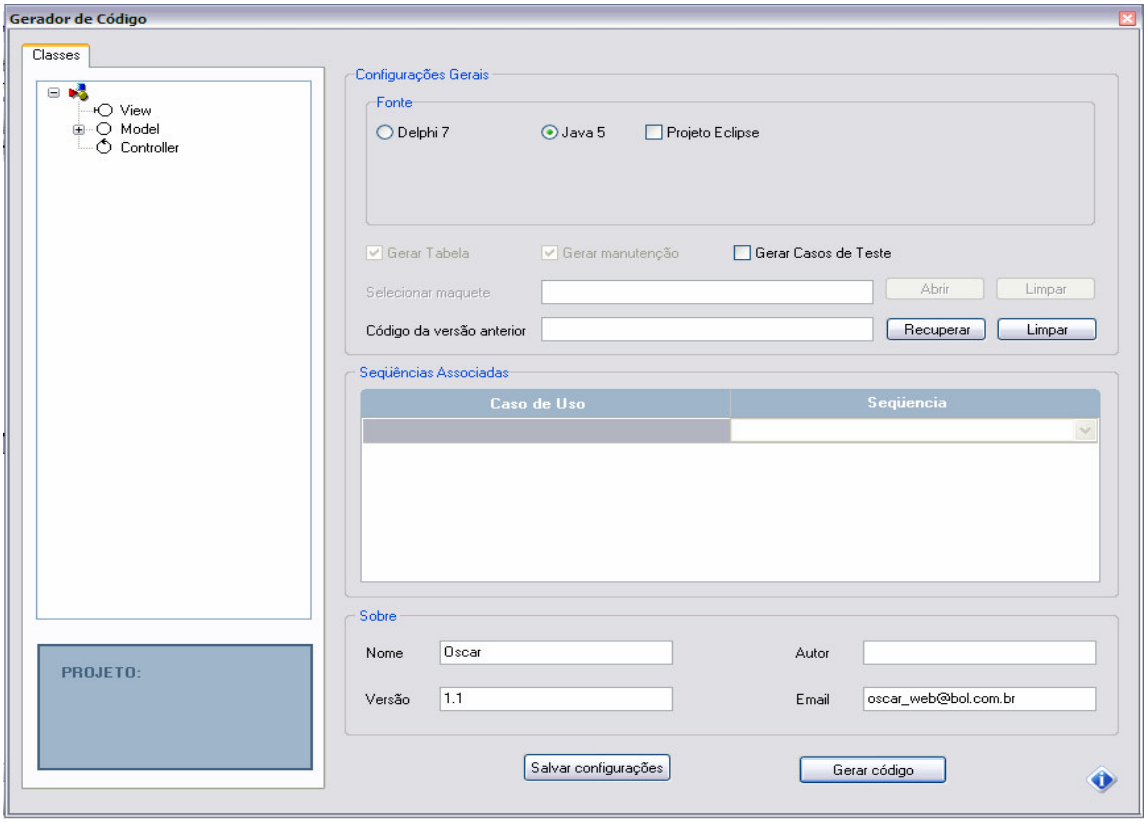

FIGURA 14 – Criando PSM e Código Fonte.

Nas Figuras 15, 16, 17 e 18 são apresentadas as telas do cadastro de cliente, de carro, de pagamento e de locação, respectivamente, sendo que a implementação foi gerada em Java pela ferramenta RAPDIS.

Na Figura 15 é ilustrada a tela de Cadastro de Cliente (inserção, alteração e remoção) com os campos código e nome.

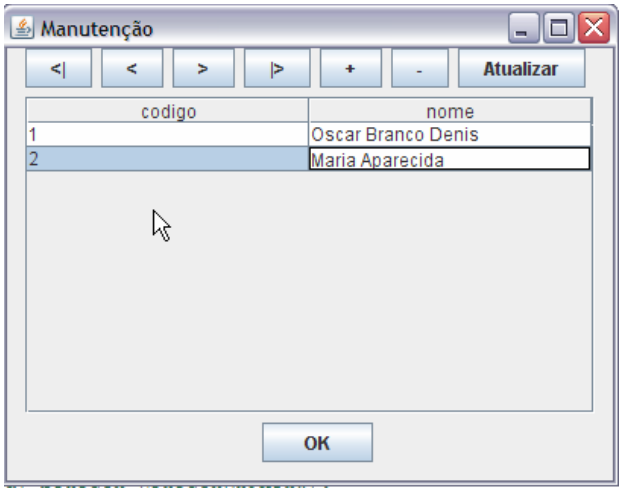

FIGURA 15 - Cadastro de Cliente

Na Figura 16 é ilustrada a tela de Cadastro de Carro com os campos placa, marca, modelo e locado.

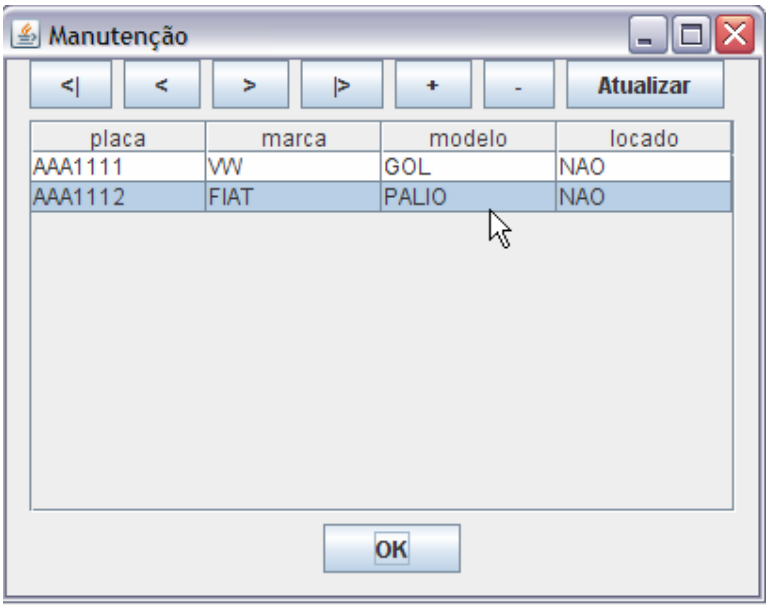

FIGURA 16 – Cadastro de Carro

Na Figura 17 é ilustrada a tela de cadastro de Pagamento com os campos código e descrição.

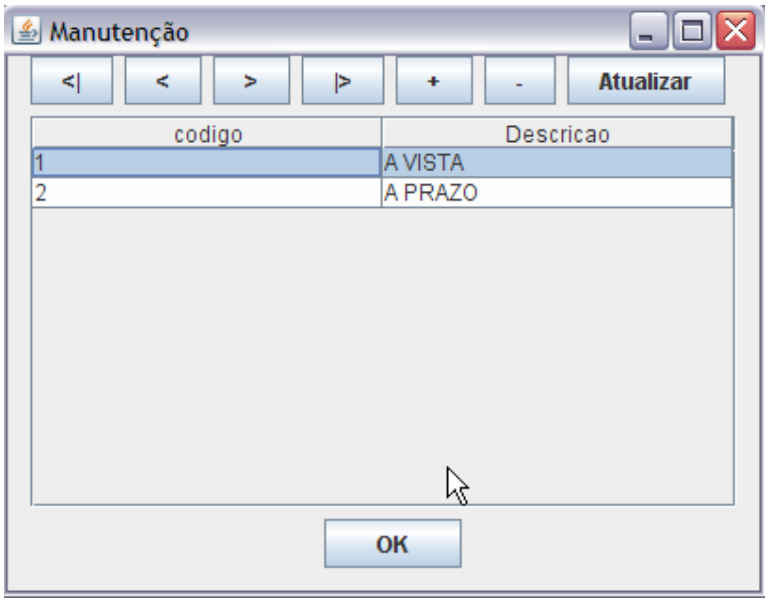

FIGURA 17 – Cadastro de Pagamento

Na Figura 18 é ilustrada a tela de Locação com os campos número, data, data devolução, carro, cliente e pagamento.

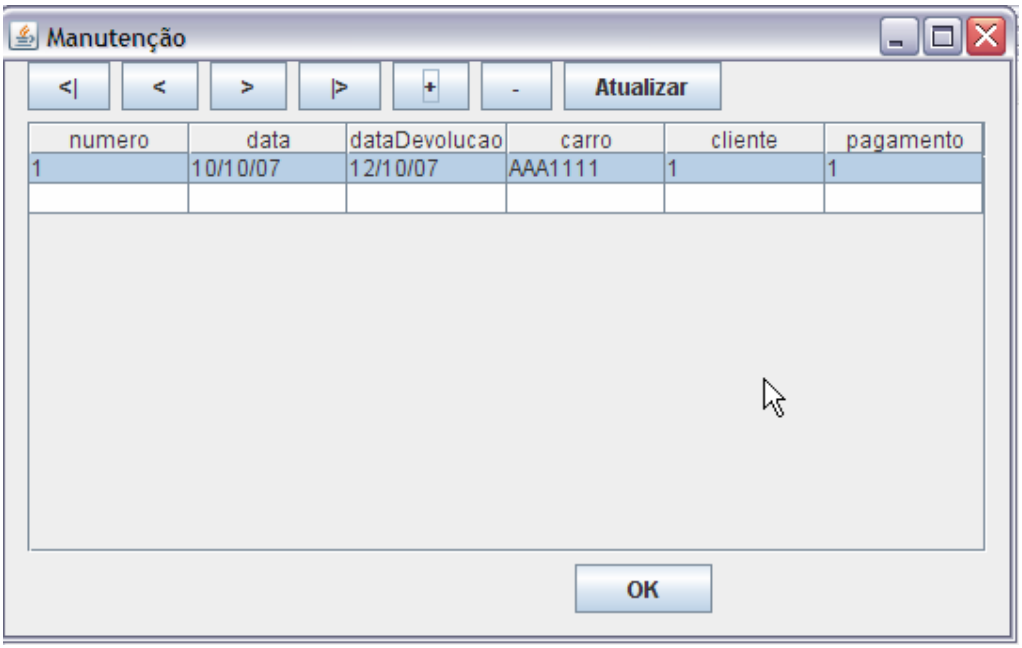

FIGURA 18 – Cadastro de Locação

Durante a execução do sistema, observou-se que a integridade referencial não foi considerada durante a geração do código fonte e das tabelas. A partir disso, optou-se pela alteração do SGBD (Sistema Gerenciador de Banco de Banco de Dados) para o MySQL uma vez que este é gratuito e amplamente utilizado tanto no meio acadêmico quanto no meio industrial. Além da alteração do SGBD foram adicionadas chaves estrangeiras nas tabelas para o tratamento de integridade referencial.

3.2 Migração do Paradox para o MySQL

Para possibilitar que o sistema funcionasse com o SGBD MySQL e considerasse integridade referencial, foram substituídas duas linhas de código no método `DomainDB()` da classe `DomainDB.java` (rótulo 1, Figura 19) para permitir conexão com qualquer SGBD por meio do ODBC (*Open Data Base Connectivity*) (rótulo 2, Figura 19). Esse método, antes de ser alterado neste trabalho, efetuava a conexão somente com o SGBD Paradox.

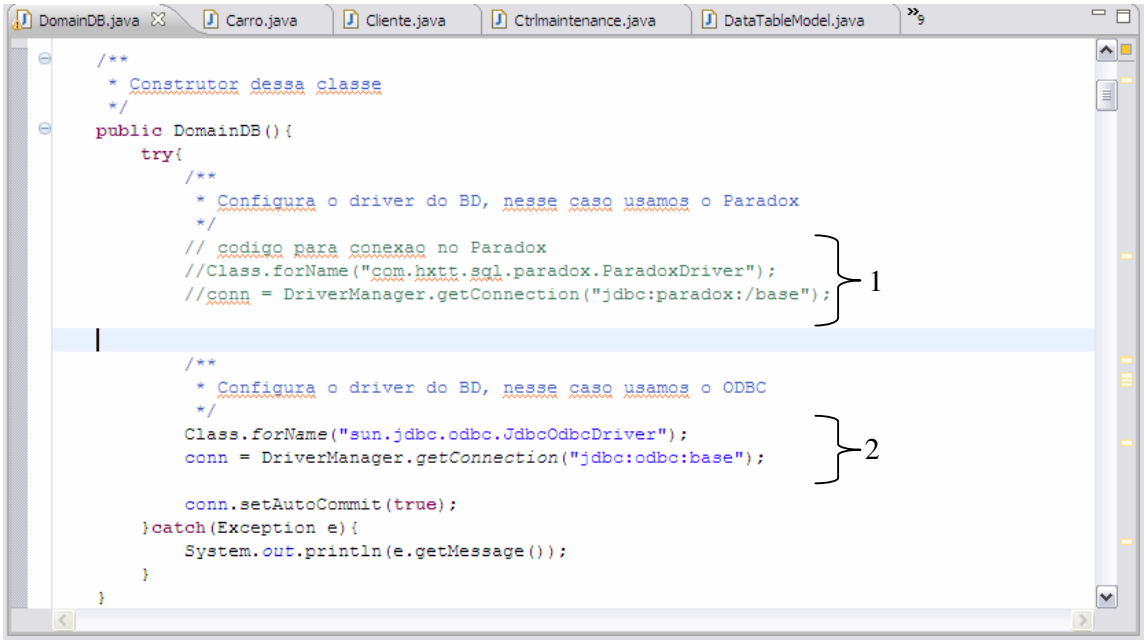

FIGURA 19 – Alteração no método `DomainDB()` da classe `DomainDB`

Para que a integridade referencial fosse considerada, foram criados *scripts* para a criação das tabelas no SGBD MySQL, levando em consideração chaves primárias e estrangeiras. Na Figura 20 é ilustrado o *script* criado para a geração da tabela Locação. Os *scripts* das demais tabelas estão apresentados no Apêndice C.

Antes de efetuar a criação das tabelas do sistema no SGBD MySQL, foi necessário criar uma base de dados chamada "BASE", tal nome foi determinado pela ferramenta RAPDIS.

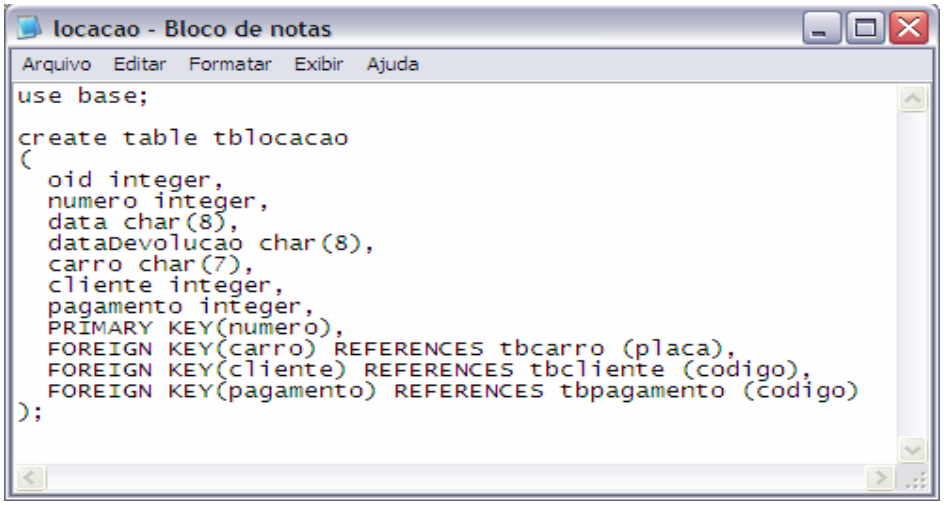

FIGURA 20 – *Script* da tabela Locação

Em seguida, o sistema foi executado e testes de inserção de registros foram realizados. Durante esses testes foram encontrados dois erros.

O primeiro erro, apresentado na Figura 21 e descrito no anexo D, está relacionado à classe `layout` do pacote `org.jdesktop`. Para resolvê-lo foi necessário copiar o arquivo *"swing-layout-1.0.jar"* no diretório bin do projeto Eclipse. Essa API (*Application Programming Interface*) é utilizada no código gerado pela ferramenta RAPDIS.

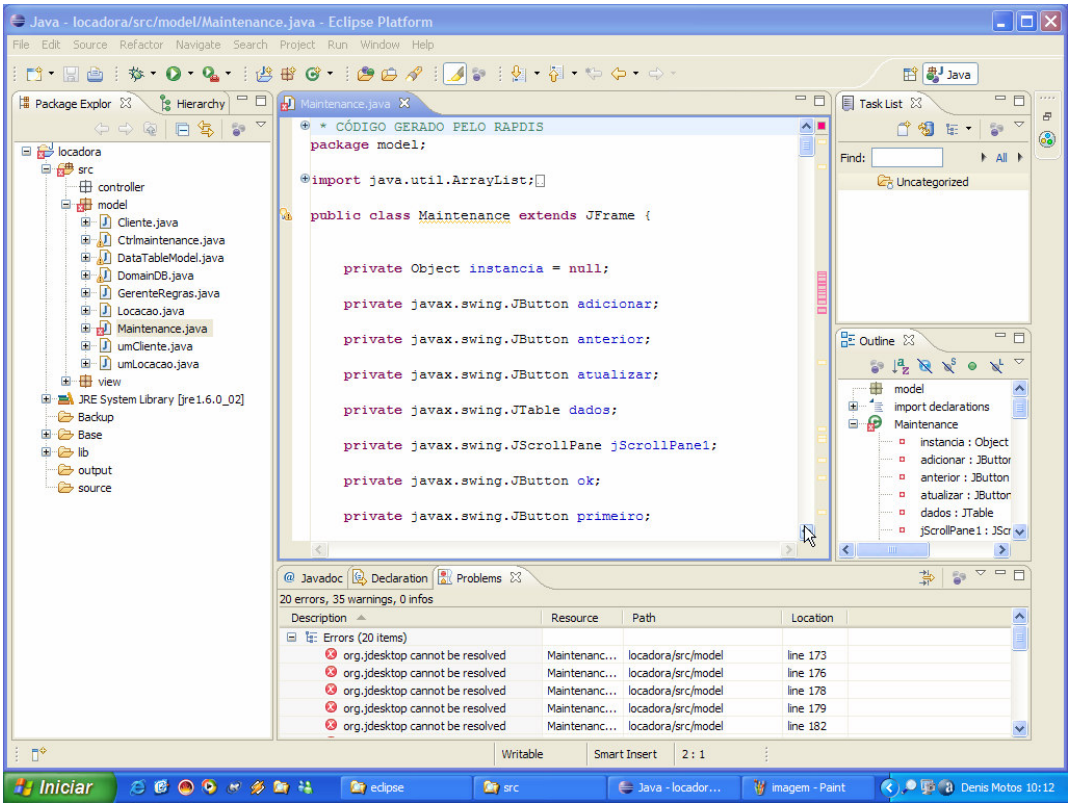

FIGURA 21 – Erro 1 - swing-layout-1.0.jar

O segundo erro, apresentado na Figura 22 e descrito no anexo E, está relacionado ao fechamento do objeto da classe `ResultSet`. Esse objeto é utilizado para efetuar a manipulação dos dados no SGBD. Para solucionar esse problema, a linha de código que gerou tal erro foi comentada, conforme apresentado no rótulo 1 da Figura 23.

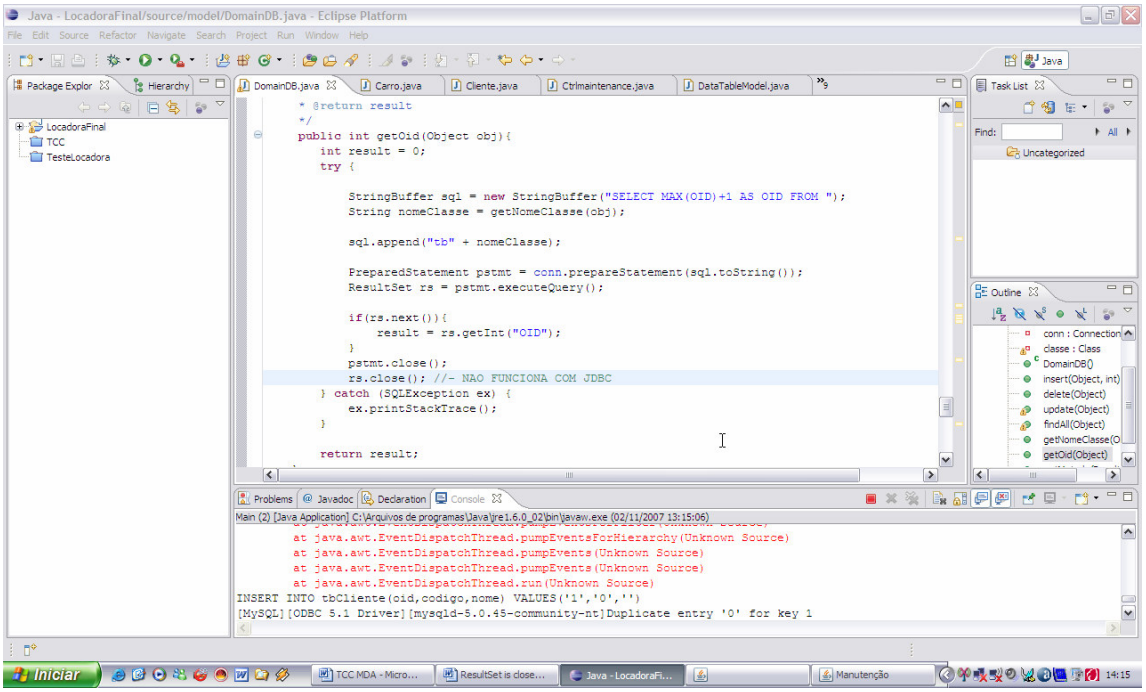

FIGURA 22 – Erro 2 – Encerramento incorreto do objeto da classe `ResultSet`

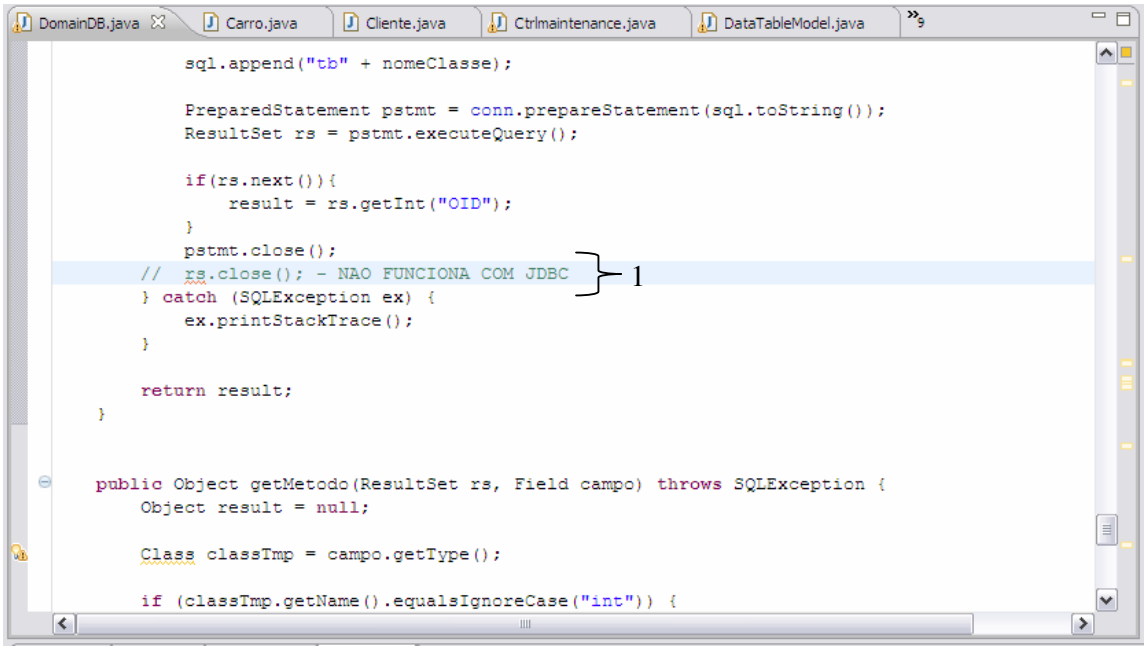

FIGURA 23 – Linha de código inutilizada

A partir do desenvolvimento do sistema de locação de veículos por meio da ferramenta RAPDIS, que é baseada em MDA, observou-se que esse novo tipo de

desenvolvimento traz diversos benefícios como é o caso do aumento da produtividade das equipes uma vez que grande parte do código fonte pode ser automaticamente gerado. No entanto, é necessário ter precaução durante o desenvolvimento incremental uma vez que alterações manuais podem ser efetuadas, sendo necessário um controle sistemático das versões do sistema sendo desenvolvido. No próximo capítulo têm-se as conclusões obtidas, as limitações do trabalho conduzido bem como sugestões de trabalhos futuros.

CONCLUSÕES

Resultados Obtidos e Contribuições

Como principais resultados e contribuições do trabalho têm-se o estudo e entendimento do desenvolvimento baseado em modelos, mais especificamente, da arquitetura baseada em modelos (MDA). Observou-se que MDA é uma nova técnica de apoio ao reúso de *software* e, consequentemente, ao desenvolvimento com qualidade em um intervalo de tempo menor, com custos reduzidos. Isso foi observado com o estudo de caso realizado com o apoio de uma ferramenta específica.

Outra facilidade apontada por diversos autores é quanto a manutenção, uma vez que esta é feita em geral no próprio modelo, permitindo que a documentação fique atualizada em relação ao código fonte. No estudo de caso realizado, as manutenções tiveram que ser realizadas no próprio código fonte uma vez que houve alteração de plataforma, no caso o SGBD, que não era suportado pela ferramenta.

Limitações do Trabalho

A ferramenta Rapdis não contém um tutorial de funcionamento, fazendo com que o projeto fosse desenvolvido na tentativa e erro.

O estudo de caso utilizando foi desenvolvido em apenas uma ferramenta, com isso não foi possível observar as vantagens e desvantagens da RAPDIS comparadas a outras ferramentas.

Como o sistema desenvolvido é de pequeno porte, não foi possível realizar o desenvolvimento de maneira iterativa e incremental e, consequentemente, não foi possível analisar os problemas que poderiam ser enfrentados com a ausência de controle de versões não providos pela ferramenta RAPDIS. Outra limitação que pode ser apontada é que somente o modelo PIM foi desenvolvido no estudo de caso.

Trabalhos Futuros

Como trabalhos futuros, decorrentes do trabalho realizado, têm-se os elencados a seguir:

- Desenvolvimento baseado em modelos do sistema de pequeno porte utilizado neste trabalho em diversas ferramentas de apoio a MDA a fim de efetuar um estudo comparativo entre elas observando características de legibilidade do código, qualidade do sistema, usabilidade, dentre outras.
- Desenvolvimento baseado em modelos utilizando o modelo de processo iterativo e incremental visando a observar e prover diretrizes para o controle das versões do código fonte gerado.
- Desenvolvimento baseado em modelos utilizando a ferramenta RAPDIS, elaborando também o modelo CIM.

REFERÊNCIAS

BELIX, José Eduardo. **Um Estudo Sobre MDA: Suporte fornecido pela UML e Reuso de Soluções Pré-Definidas**. 2006. Dissertação (Mestrado em Engenharia Elétrica) Departamento de Engenharia da Computação e Sistemas Digitais, Escola Politécnica da Universidade de São Paulo, São Paulo, 2006.

COSTA, Cláudio, COELHO, Diogo, OLIVEIRA, Fábio, SILVA, Marcos, VILAÇA, Susana. **Model Driven Development.** <http://paginas.fe.up.pt/~aaguiar/es/artigos%20finais/es_final_5.pdf> Acesso em 09 abril 2007.

DEBONI, José Eduardo Zindel. **Breve Introdução aos Diagramas da UML. <** http://www.voxxel.com.br/pages/introdiauml.html > Acesso em 29 abril 2007.

FERREIRA, João Paulo Augusto de Oliveira. **XO2 – Um Gerador de Código MDA Baseado em Mapeamento de Modelos**. 2005. Dissertação (Mestrado em Ciência da Computação) Centro de Informática da Universidade Federal de Pernambuco, Universidade Federal de Pernambuco, Pernambuco, 2005.

GAMMA, E.; HELM, R.; JOHNSON, R.; VLISSIDES, J. **Design patterns elements of reusable of object-oriented software**. second ed. Addison-Wesley, 1995.

KRIESER. **MDA – Model Driven Architecture & UML – Unified Modeling Language <**http://www.krieser.com.br/?on=11&in=8> Acesso em 26 março 2007.

MAIA, Natanael E. N. **Odyssey-MDA: Uma Abordagem Para a Transformação de Modelos**. 2006. Dissertação (Mestrado em Ciência da Computação) Coordenação dos Programas de Pós-Graduação, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2006.

MORGADO, Gisele P. **RAPDIS: UM PROCESSO MDA PARA DESENVOLVIMENTO DE SISTEMA DE INFORMAÇÃO.** 2006. Artigo Sipósio Brasileiro de Sistema de Informação. Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2007.

MORGADO, Gisele Pereira. **RAPDIS: UM PROCESSO E UM AMBIENTE MDA PARA O DESENVOLVIMENTO DE SISTEMAS DE INFORMAÇÃO.** 2007. Dissertação (Mestrado em Informática) Instituto de Matemática Núcleo de Computação Eletrônica, Universidade Federal do Rio de Janeiro, Rio de Janeiro, 2007.

OMG Object management group. Disponível em :<http://www.omg.org/mda/>Acesso em: 04 abril 2007.

SANTOS, Gustavo, SAMPAIO, Augusto. **Integrando Sistemas de Transformação de Programas e Metodologias Ágeis. <**http://www.cin.ufpe.br/~gas/papers/STS_MetAgeis.pdf**>** Acesso em 01 abril 2007.

ANEXO A - Sistema para Locadora de Carros

A – VISÃO GERAL DO SISTEMA

O sistema para a Locadora de Carros consiste do gerenciamento dos aluguéis de carros. O sistema deve ainda emitir algumas consultas, possibilitando um melhor gerenciamento dos aluguéis.

B – REQUISITOS FUNCIONAIS

B1 – Lançamentos diversos

1. O sistema deve permitir a inclusão, alteração e remoção de clientes da locadora de carros, com os seguintes atributos: Código e nome.
2. O sistema deve permitir a inclusão, alteração e remoção dos carros para aluguel pertencentes à locadora de carros. Cada carro possui os seguintes atributos: placa do carro, marca, modelo e se o carro está locado ou não. Para cada categoria de carro podem existir diversos carros com placas, modelos e anos diferentes.
3. O sistema deve permitir a retirada de carro por um cliente. Cada retirada de carro possui os seguintes atributos: data da retirada do carro, data da devolução do carro, identificação do cliente (previamente cadastrado), placa do carro alugado.
4. O sistema deve permitir a devolução do carro por um cliente, com os seguintes atributos: placa do carro alugado, data da devolução do carro, número de diárias cobradas, valor de cada diária.
5. O sistema deve permitir as seguintes opções de pagamento do aluguel do carro: 1) à vista (em dinheiro, cheque ou cartão de crédito); 2) faturado em 30 dias.

B2 – Impressão de diversos tipos de consultas

6. O sistema deve permitir a impressão de uma listagem dos carros alugados no momento, contendo o nome do cliente, placa, marca e modelo do carro, data de retirada e data prevista para devolução.
7. O sistema deve permitir ao cliente imprimir um histórico de seus aluguéis de carro. Para tal o cliente deve ter sido previamente cadastrado e deve portar um código de identificação e uma senha. Esse histórico deve conter uma linha para cada carro alugado pelo cliente, contendo as datas de retirada e devolução e os totais pagos em cada ocasião.

C – REQUISITOS NÃO FUNCIONAIS

C1. Eficiência

8. O sistema deve responder a consultas *on-line* em menos de 5 segundos.

C2. Portabilidade

9. O sistema deve ser executado em computadores Pentium 200mHz ou superior, com sistema operacional Windows 98 ou acima.
10. O sistema deve ser capaz de armazenar os dados em base de dados MySQL.

ANEXO B – Manual de Instalação e Uso da Ferramenta RAPDIS

Neste anexo apresenta-se informações com respeito à aquisição e instalação da ferramenta acadêmica RAPDIS, bem como um passo a passo de como utilizar a mesma.

B1.Aquisição e Instalação da Ferramenta

Para adquirir a ferramenta RAPDIS é necessário fazer download no endereço (http://www.geti.dcc.ufrj.br), menu projetos, opção RAPDIS. Para efetuar a instalação da ferramenta, é necessário seguir alguns passos, como por exemplo, indicar o diretório em que a ferramenta será instalada. Depois da instalação da ferramenta é gerado um atalho do executável do RAPDIS no desktop. Para que sejam criadas as tabelas do banco de dados da aplicação gerada por meio da ferramenta, é necessário a instalação do BDE (BORLAND DATABASE ENGINE) caso a linguagem Delphi não esteja instalada.

B2. Elaboração do Projeto

Para o desenvolvimento de um sistema na ferramenta RAPDIS será utilizado um sistema pequeno porte de locação de carros. Esse sistema tem como objetivo efetuar o controle de aluguel de carros. O documento de requisitos do sistema está no Anexo A.

B2.1. Passos da utilização da RAPDIS

Para iniciar a utilização da ferramenta é necessário selecionar um projeto existente ou criar um novo, conforme apresentado na Figura B1.

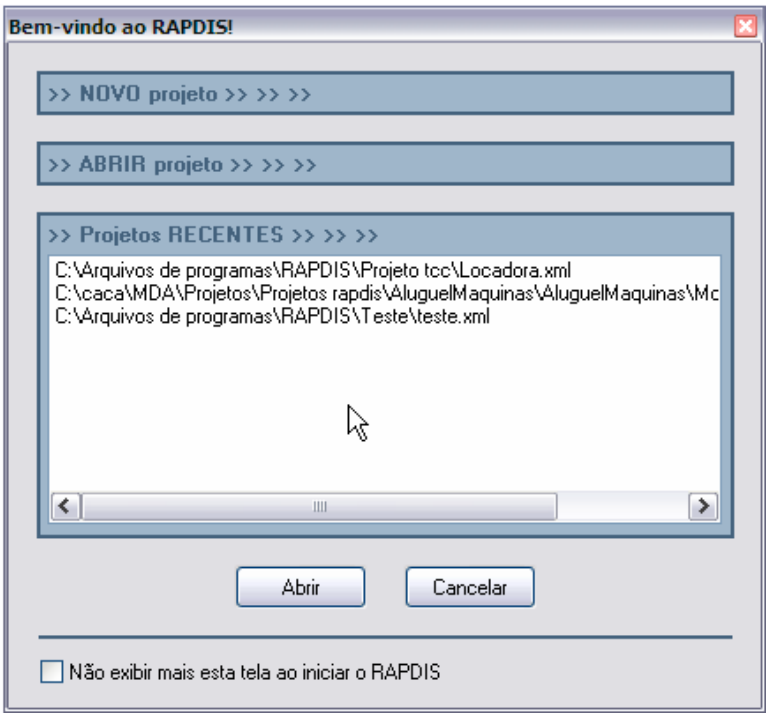

Figura B1 – Tela inicial da ferramenta RAPDIS

A partir da escolha da opção “NOVO projeto” é apresentada uma nova tela chamada “Novo Projeto”, conforme ilustrado na Figura B2. Nessa janela são apresentadas as opções para que o desenvolvedor possa escolher o local onde o projeto vai ser armazenado e informar o nome que será atribuído ao projeto.

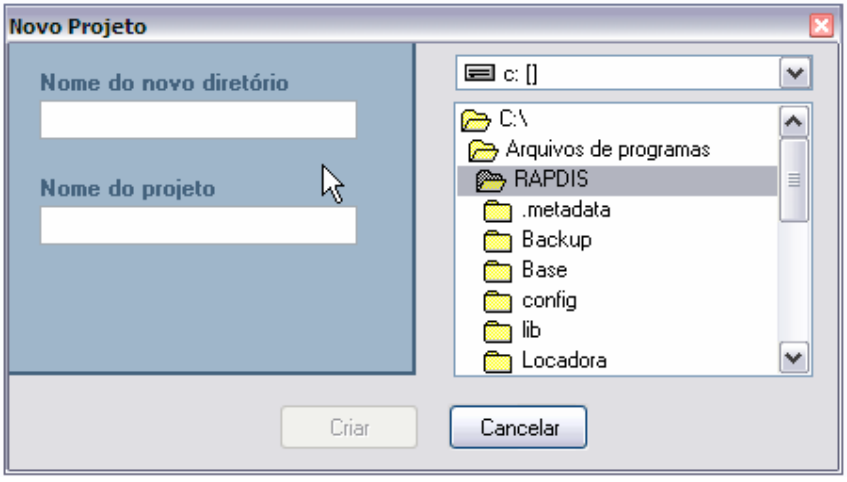

Figura B2 – Tela para criar um novo projeto

B2.1.1. Elaboração do PIM

A elaboração do PIM, por meio do diagrama de classes utilizando notação UML, foi efetuada na própria ferramenta RAPDIS, ou seja, a própria ferramenta fornece interface gráfica para a criação do diagrama de classes. Para isso é necessário selecionar o módulo da ferramenta “Modelo de Sistema de Informação”, por meio da aba com mesmo nome, como apresentado na Figura B3.

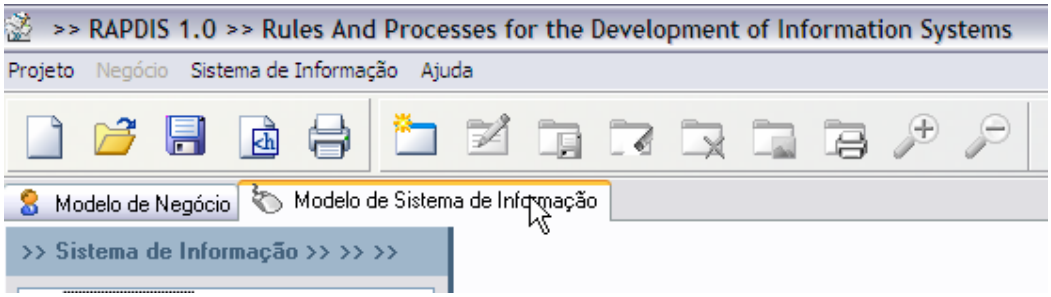

Figura B3 – Seleção do módulo Modelo de Sistema de Informação

Para a criação do diagrama de classes, que representa o PIM, é necessário selecionar o menu “Sistema de Informação”, opção “Novo diagrama ...”, conforme ilustrado na Figura B4. A partir disso uma nova tela é apresentada ao desenvolvedor para selecionar o tipo do diagrama que deseja criar e para informar um nome para o mesmo, como apresentado na Figura B5.

.

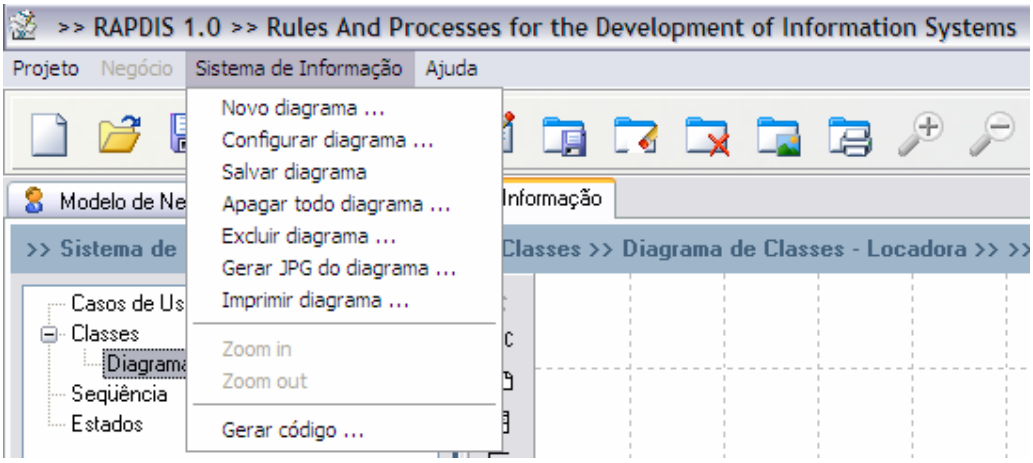

Figura B4 – Criando Novo Diagrama

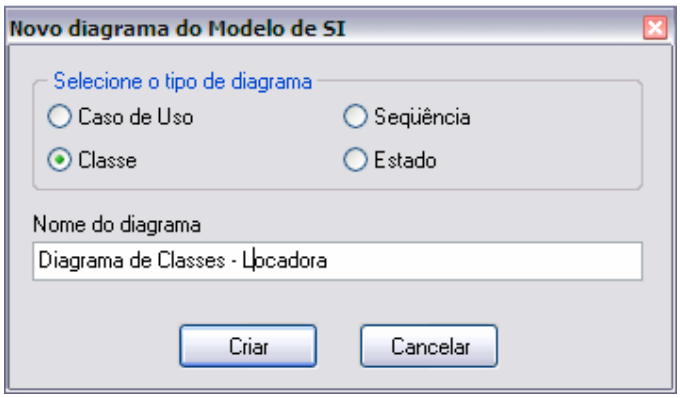

Figura B5 – Criando Diagrama de Classes (PIM)

Em seguida, para o projeto da Locadora de Carros foi criada a classe "Cliente" utilizando a barra de ferramentas disponibilizada pela ferramenta, como ilustrado na Figura B6.

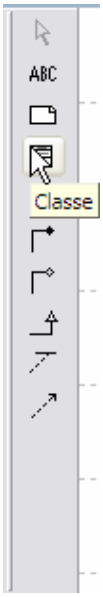

Figura B6 – Barra de Ferramentas.

Durante a criação das classes do diagrama, o desenvolvedor deve classificá-las como Classe de Domínio, de Interface ou de Controle. Isso é feito por meio da tela ilustrada na Figura B7, em que o desenvolver informa o nome da classe, bem como o seu tipo.

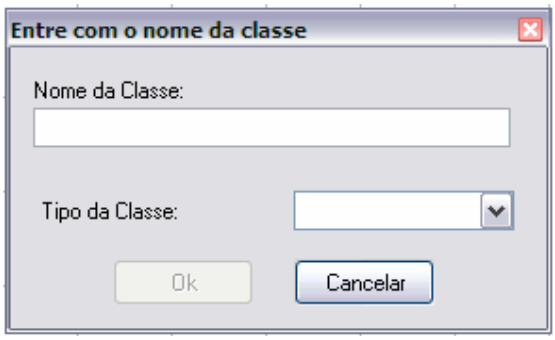

Figura B7 – Nomeando as Classes.

As classes que foram criadas estão descritas na Tabela B1, obedecendo cada uma o seu tipo.

Tabela B1 – Dados para a Criação das Classes

Nome da Classe	**Tipo de Classe**
Cliente	Domínio
Carro	Domínio
Pagamento	Domínio
Locação	Domínio

Com as Classes já criadas é necessário a inserção dos atributos para que a ferramenta futuramente crie os campos das tabelas da aplicação. Para isso, é necessário clicar com o botão direito do mouse sobre a classe e escolher a opção “Definição...”, conforme apresentado na Figura B8.

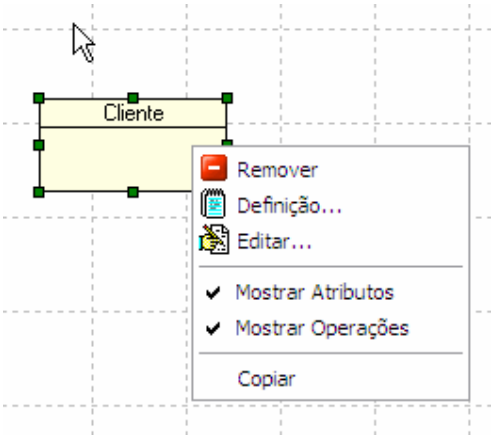

Figura B8 – Definição dos atributos.

Em seguida, a tela Definição Classe (nome_da_classe) é apresentada ao desenvolvedor, sendo necessário escolher a opção “Atributos” e, sem seguida, clicar com o botão direito do mouse e selecionar a opção ”Adicionar”, como apresentado na Figura B9.

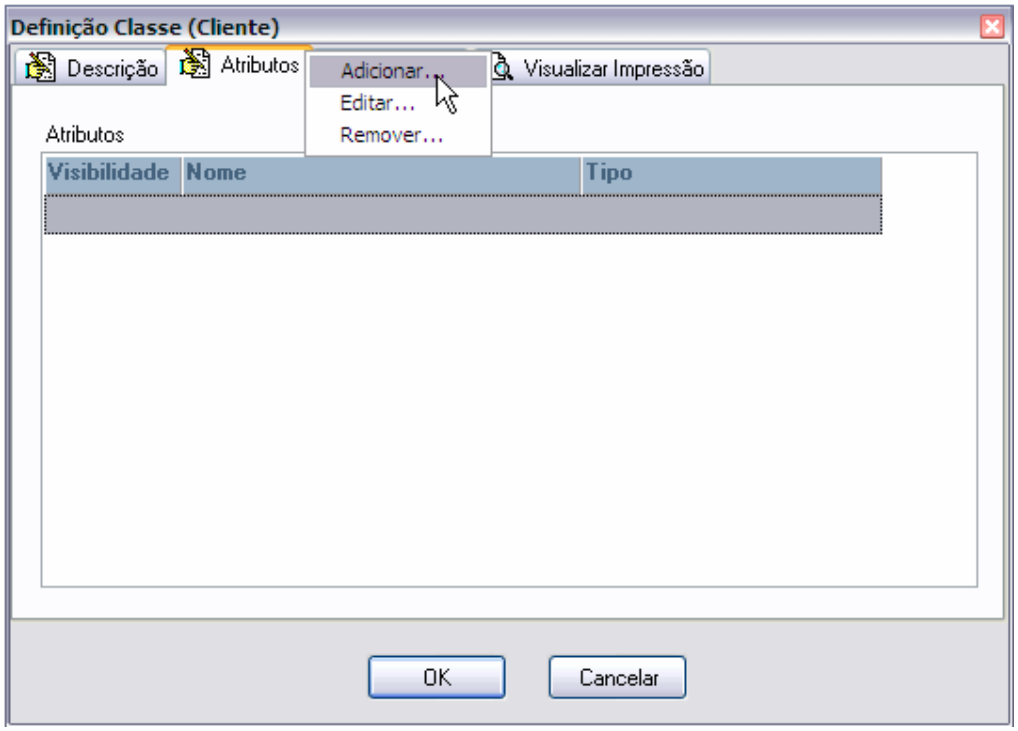

Figura B9 – Adicionar Atributos.

Posteriormente, aparecerá a janela “Definição do Atributo”, em que é digitado o nome do atributo e é selecionado a sua visibilidade (público, privado ou protegido) e o seu tipo, como ilustrado na Figura B10.

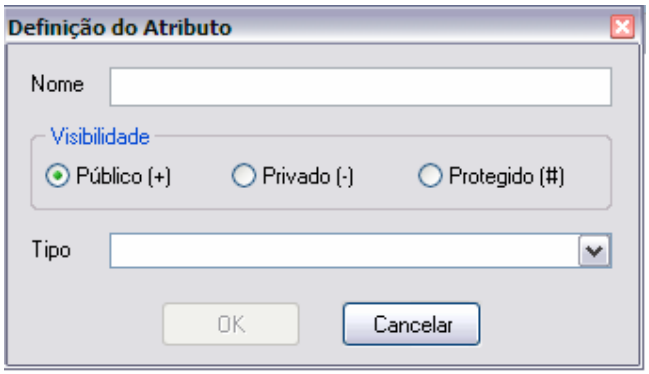

Figura B10 – Definição do Atributo.

Os atributos criados para cada classe do sistema de locação de carros são os descritos nas Tabelas B2, B3, B4 e B5.

Tabela B2 - Atributos da Classe Cliente

Nome	**Visibilidade**	**Tipo**
Codigo	Privado	Integer
Nome	Privado	String

Tabela B3 - Atributos da Classe Carro

Nome	**Visibilidade**	**Tipo**
Placa	Privado	String
Marca	Privado	String
Modelo	Privado	String
Locado	Privado	String

Tabela B4 - Atributos da Classe Pagamento

Nome	**Visibilidade**	**Tipo**
Código	Privado	Integer
Descrição	Privado	String

Tabela B5 - Atributos da Classe Locação

Nome	Visibilidade	Tipo
Numero	Privado	Integer
Data	Privado	String
Data Devolução	Privado	String
Cliente	Privado	Integer
Carro	Privado	String
Pagamento	Privado	Integer

Na Figura B11 é apresentada a representação do diagrama de classes parcial após a criação das classes e dos atributos.

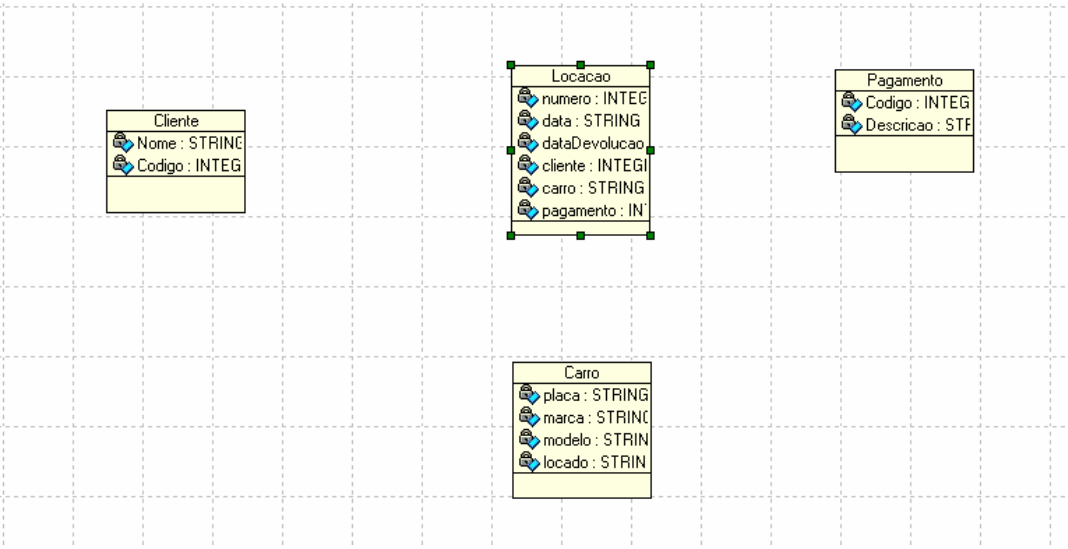

Figura B11 – Diagrama de Classes Parcial - Classes.

Para que a ferramenta associe as classes é necessário fazer as ligações, utilizando novamente a barra de ferramenta, selecionando a opção associação, como ilustrado na Figura B12.

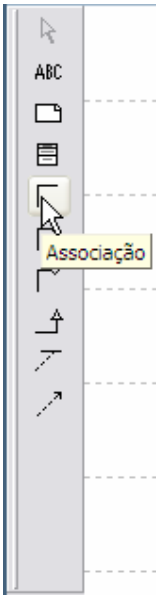

Figura B12 – Barra de Ferramenta.

Após clicar na opção “Associação” é necessário fazer a associação entre as classes desejadas clicando primeiro em uma classe e depois na outra.

Em seguida, foram estabelecidos os relacionamentos de associação entre as classes, conforme apresentado na Figura B13.

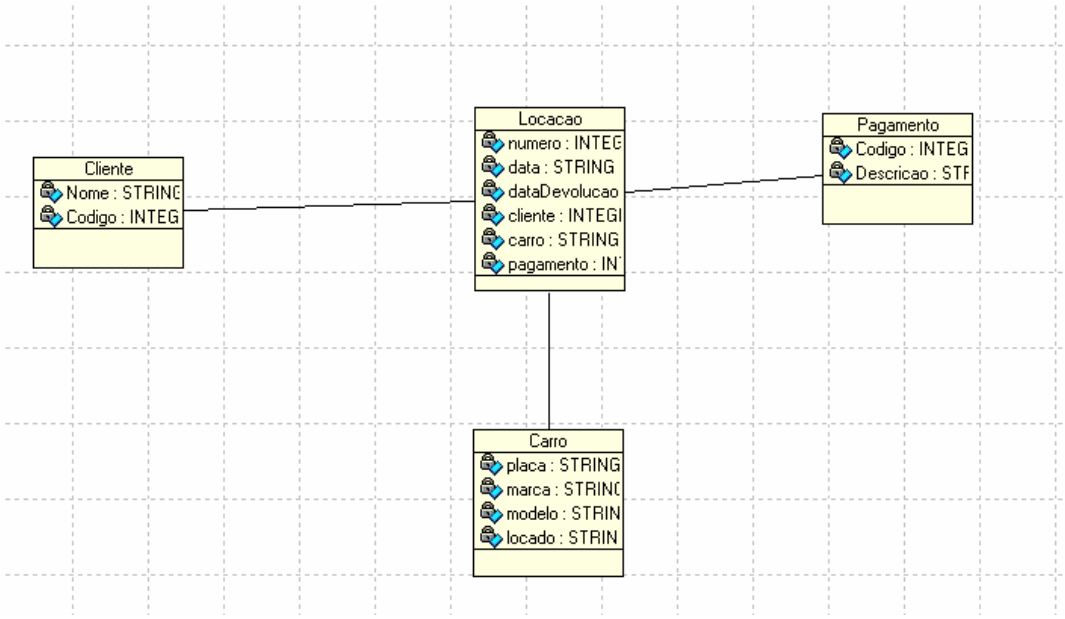

Figura B13 – Diagrama de Classes Parcial – Classes e Relacionamentos.

Após a criação das associações entre as classes é necessário efetuar a definição das mesmas. Para isso, basta clicar com o botão direito no mouse na associação desejada e escolher a opção “Definição...”, como apresentado na Figura B14.

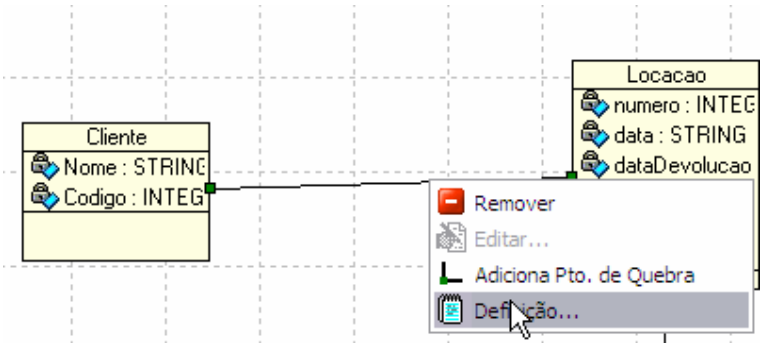

Figura B14 – Definindo Associações

Posteriormente, foram especificadas as multiplicidades e os nomes de cada relacionamento de associação definido anteriormente, conforme apresentado nas Tabelas B6, B7, B8. Tais especificações são realizadas por meio da tela “Definição Associação” ilustrada na Figura B15. Na Figura B16 apresenta-se o diagrama de classes com as associações especificadas.

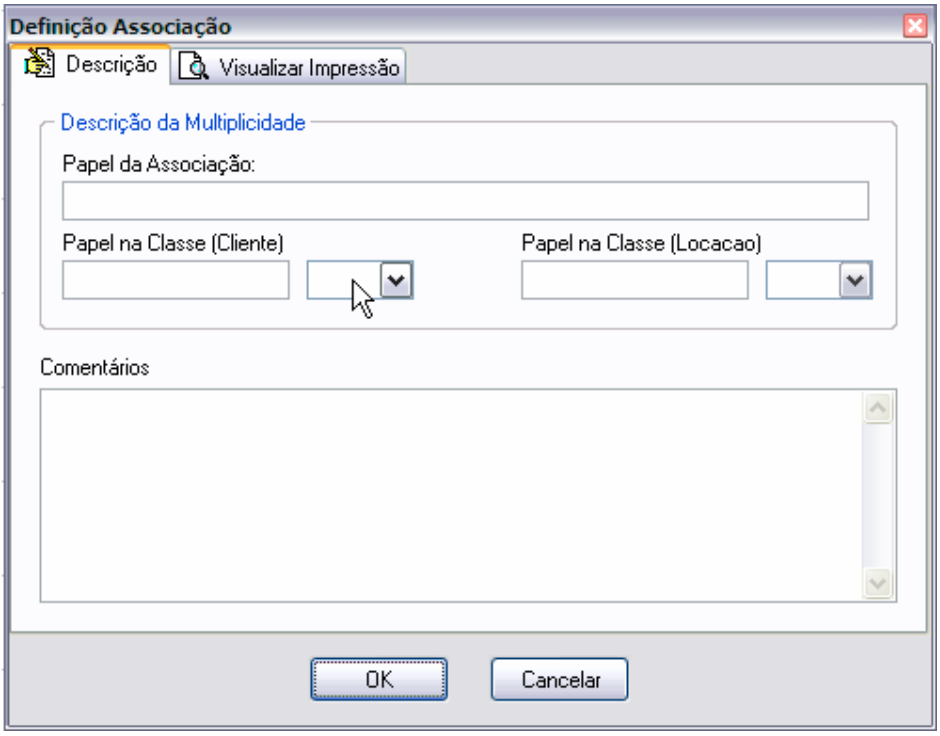

Figura B15 – Definição de Associação

Tabela B6 - Relacionamento entre as classes Cliente e Locação

Papel da Associação	Papel na Classe (Cliente) Multiplicidade	Papel na Classe (Locação) Multiplicidade
Efetua	1..1	1..*

Tabela B7 - Relacionamento entre as classes Carro e Locação

Papel da Associação	Papel na Classe (Carro)	Papel na Classe (Locação)
Possui	0..*	1..1

Tabela B8 - Relacionamento entre as classes Pagamento e Locação

Papel da Associação	Papel na Classe(Pagamento)	Papel na Classe (Locação)
Tem	0..*	1..1

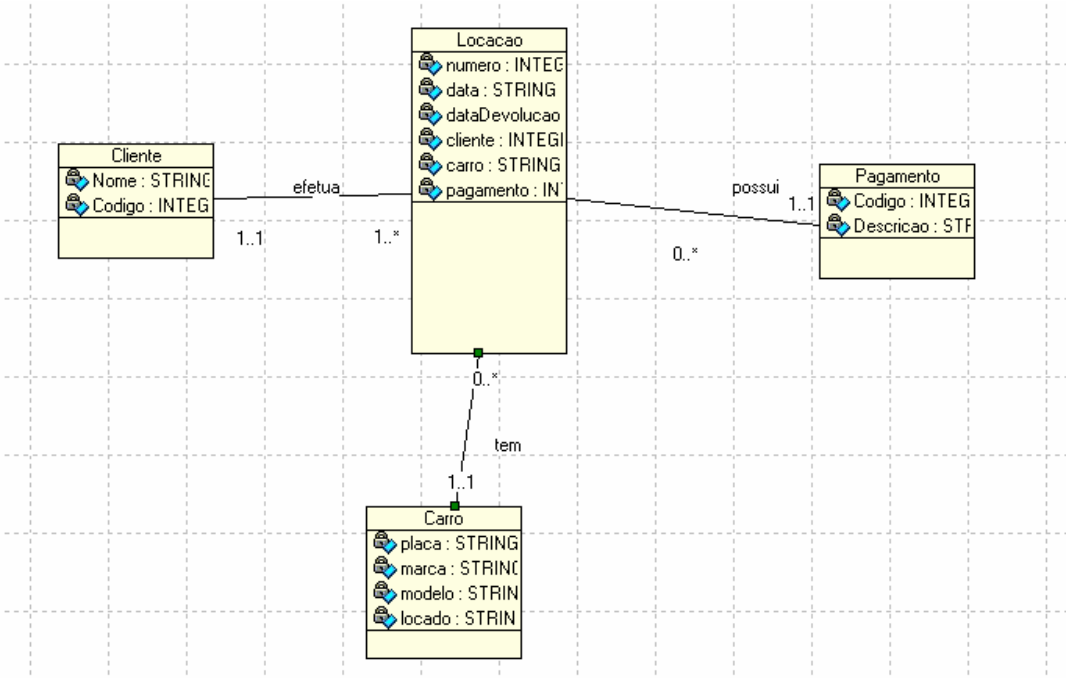

Figura B16 – Diagrama de classes completo

B2.1.2. Geração do PSM e do Código Fonte

Para a geração do PSM e criação do código fonte basta o desenvolvedor clicar no botão “Gerador de Código”, como ilustra a Figura B17.

Figura B17 – Botão Gerador de Código

Em seguida a ferramenta oferece uma janela com três opções para a criação deste modelo. Na primeira opção a ferramenta cria um PSM para a plataforma Delphi 7.0, na segunda opção a ferramenta cria um PSM para a plataforma Java 5 e na terceira a ferramenta cria um PSM para a plataforma Java com o projeto para implementar no Eclipse, como ilustrado na Figura B18. Em todos os casos, o banco de dados do sistema é gerado no Paradox.

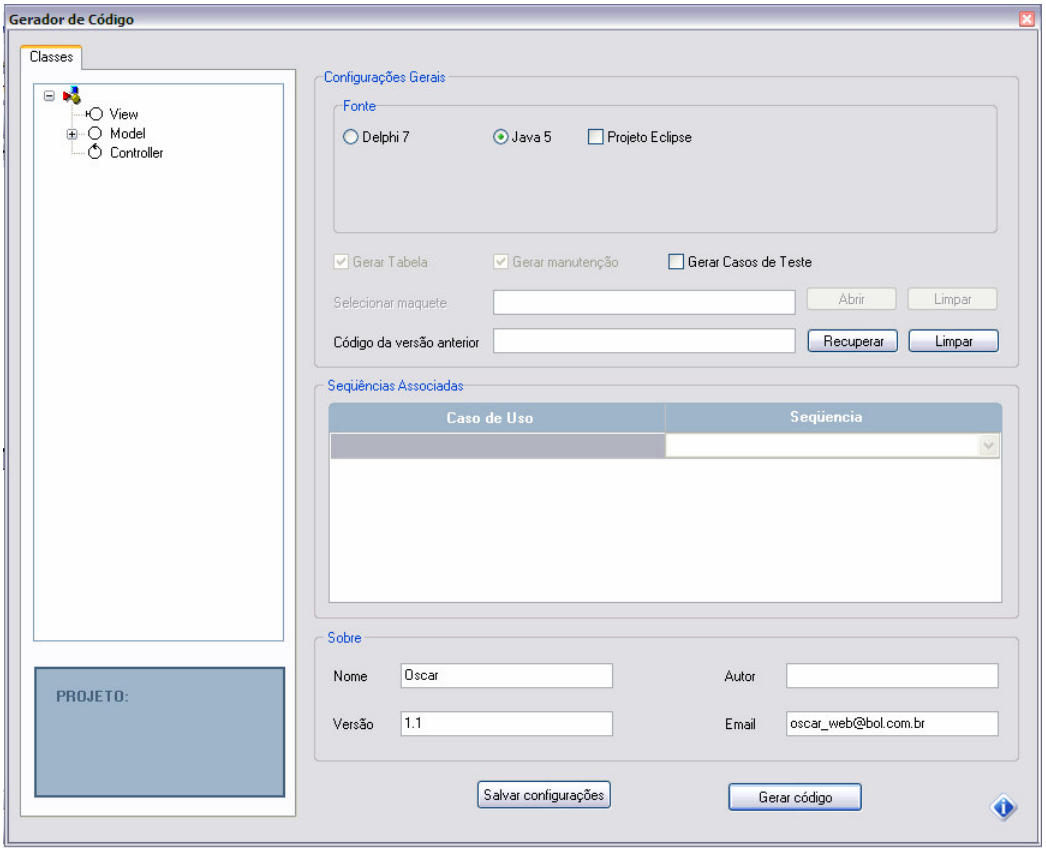

Figura B18 – Criação do PSM e do código fonte

As Figuras B19, B20, B21 e B22 apresentam o cadastro de cliente, de carro, de pagamento e de locação respectivamente.

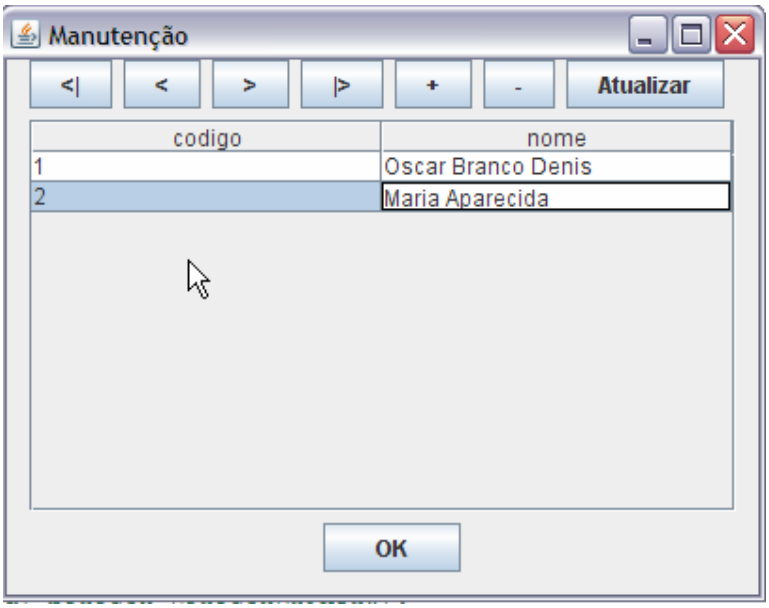

Figura B19 - Cadastro de Cliente

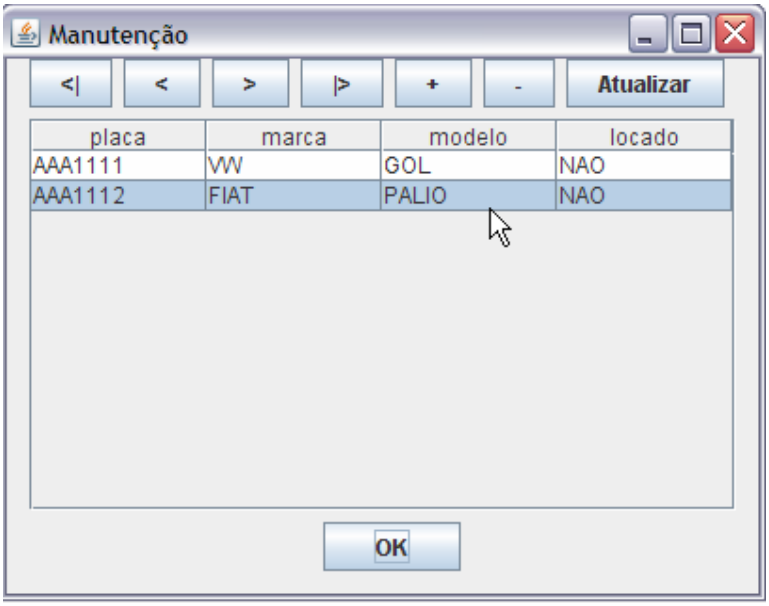

Figura B20 – Cadastro de Carro

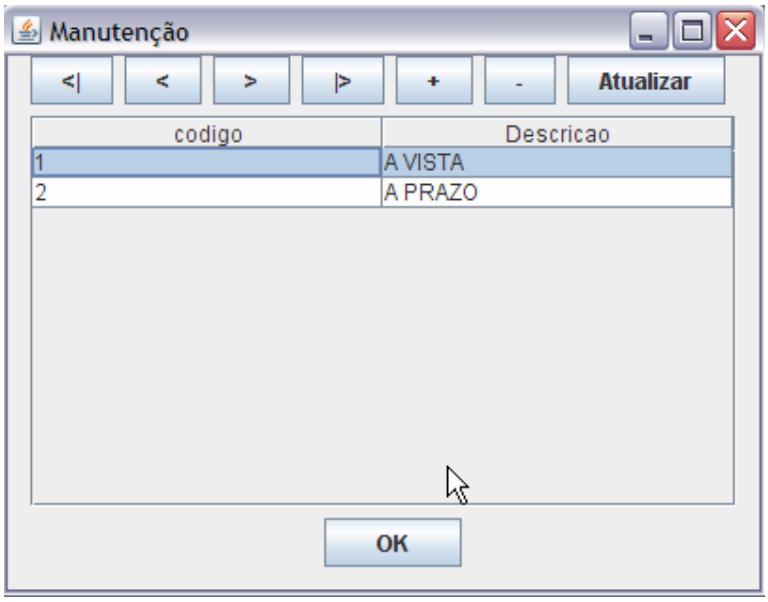

Figura B21 – Cadastro de Pagamento

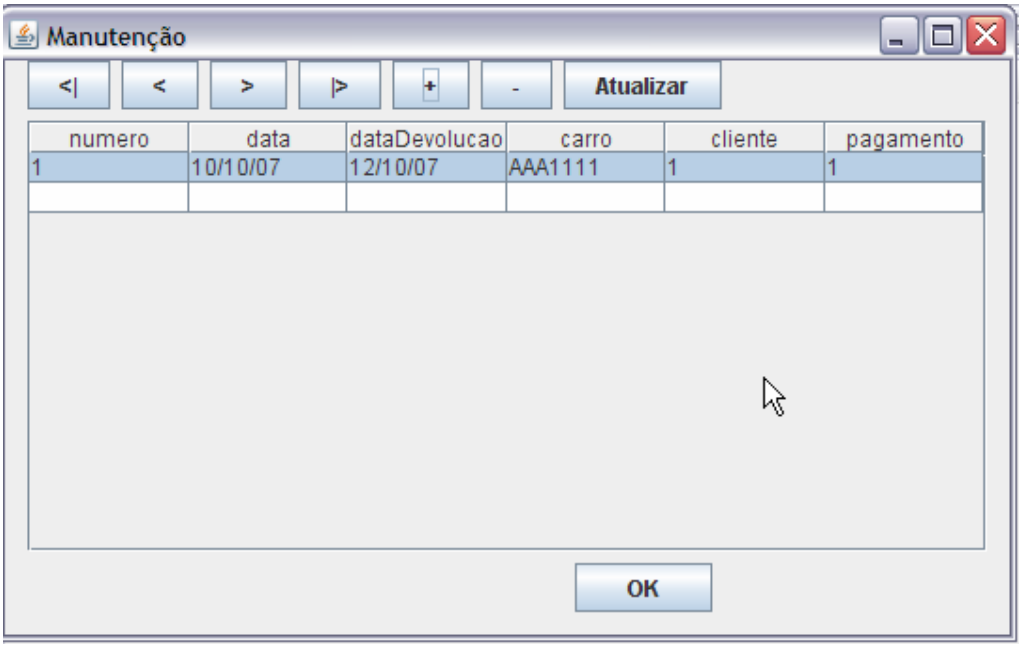

Figura B22 – Cadastro de Locação

ANEXO C – Criação do Bando de Dados do Sistema de Locação no Mysql

O objetivo deste anexo é apresentar o passo a passo para criação do banco de dados do sistema e das suas respectivas tabelas no SGBD MySQL.

Para criar o banco de dados no MySQL, primeiro é necessário abrir o MS-DOS e no prompt ir até o diretório em que o MySQL.

Primeiramente, entre no MySQL (Figura C1) e crie a Base de Dados, conforme apresentado na Figura C2.

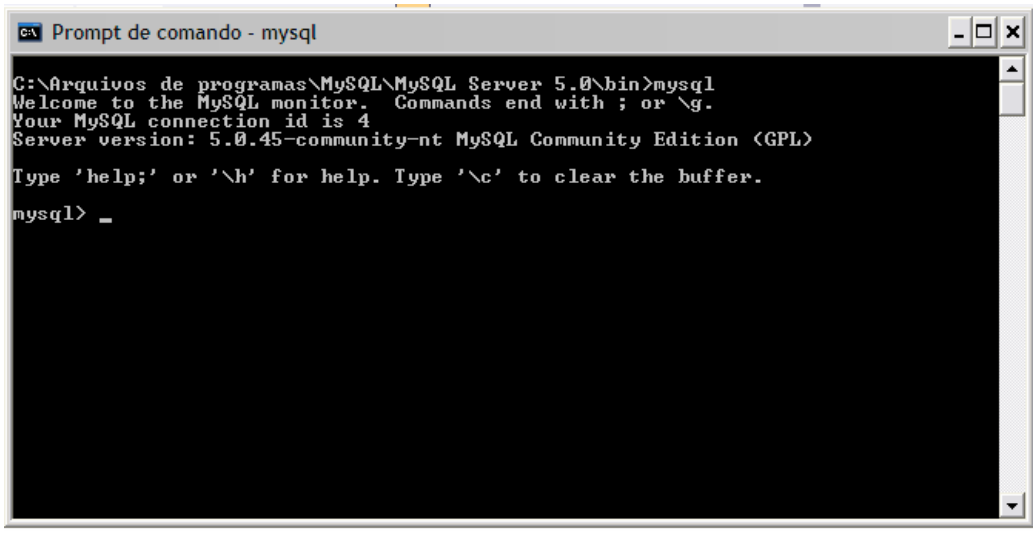

Figura C1 - Entrando no Mysql.

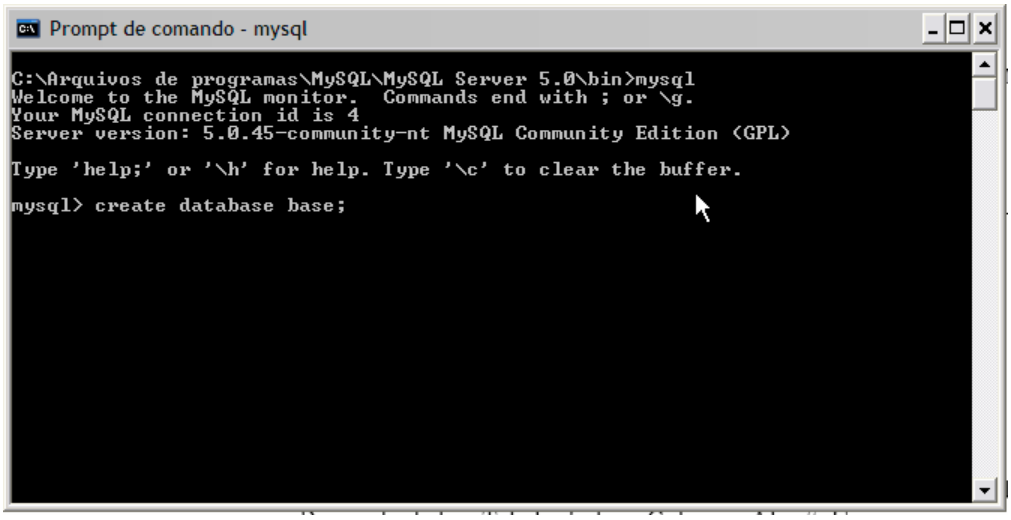

Figura C2 - Criação da Base de dados

Após criar a base de dados é necessário criar cada uma das tabelas. Para facilitar esse processo, é possível criar scripts com a definição das tabelas e em seguida executar no SGBD. Para isso, crie um arquivo com a especificação de todas as tabelas ou um para cada tabela e salve o arquivo com extensão txt ou sql no diretório \bin do MySQL

Nas Figuras C3 a C6 são apresentados os scripts de cada tabela em arquivos separados.

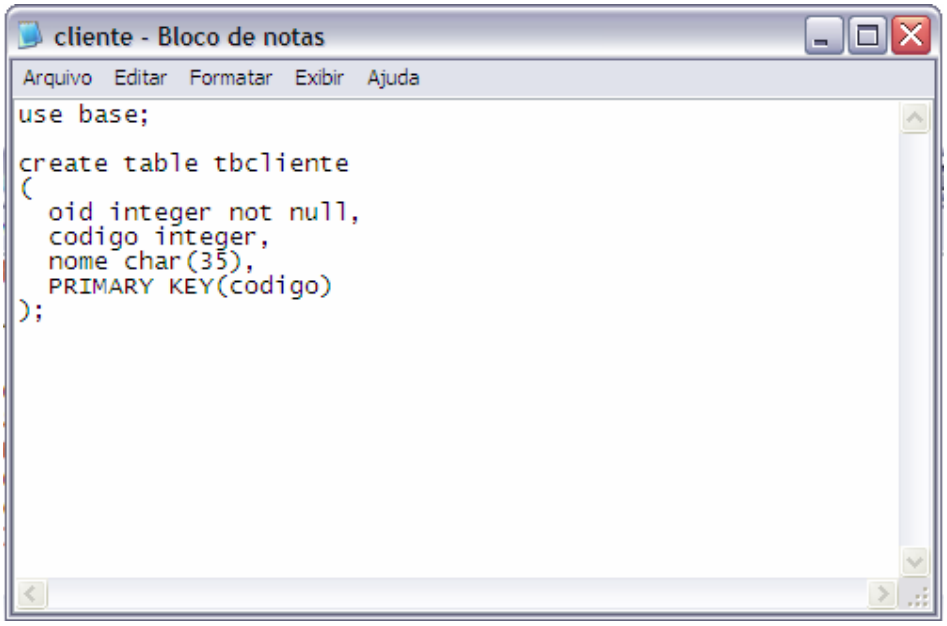

Figura C3 – Script da tabela Cliente

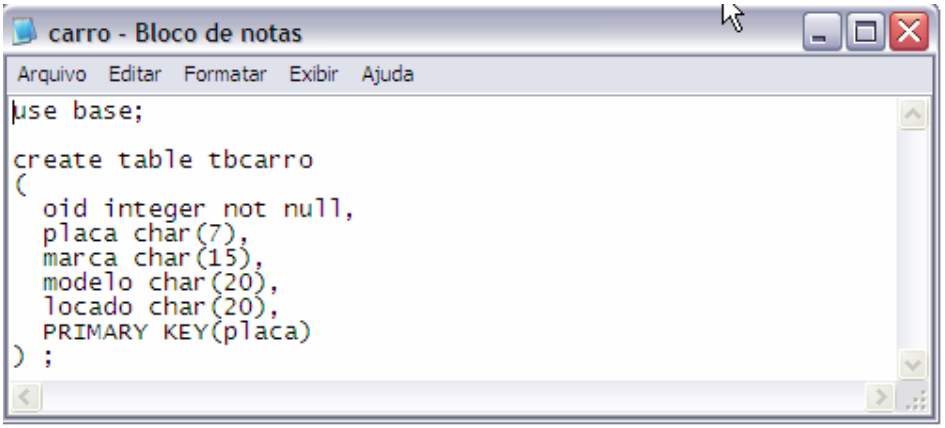

Figura C4 – Script da tabela Carro

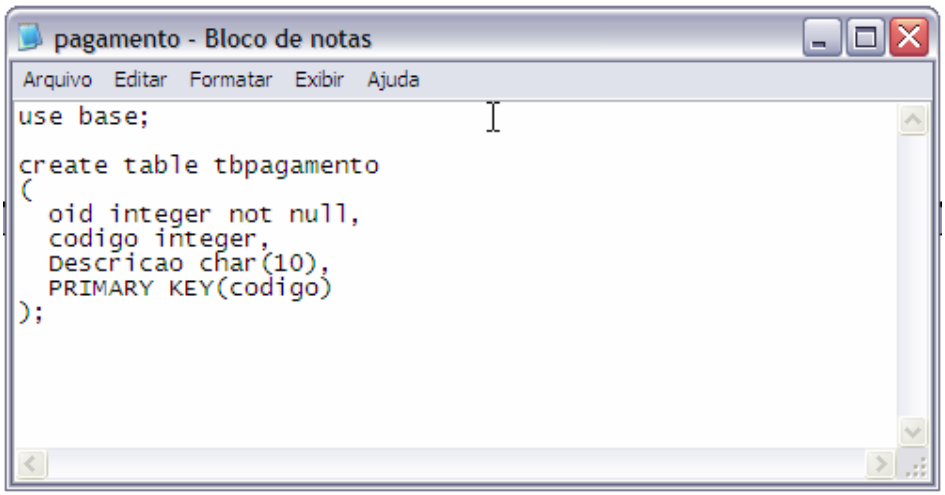

Figura C5 – Script da tabela Pagamento

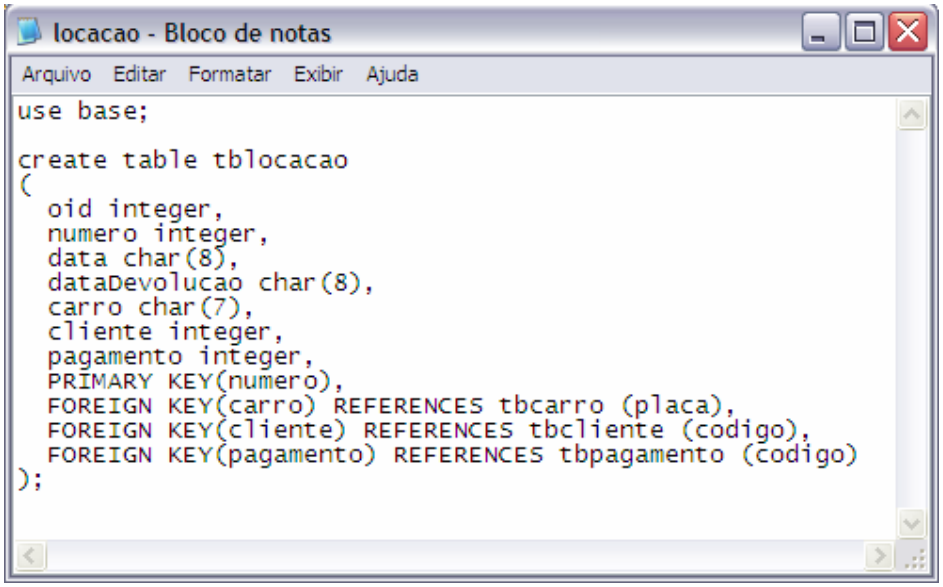

Figura C6 – Script da tabela Locação

Para executar os scripts das tabelas é necessário entrar no prompt do MS-DOS e digitar o comando “mysql < nome_do_arquivo_script.sql”, conforme apresentado na Figura C7.

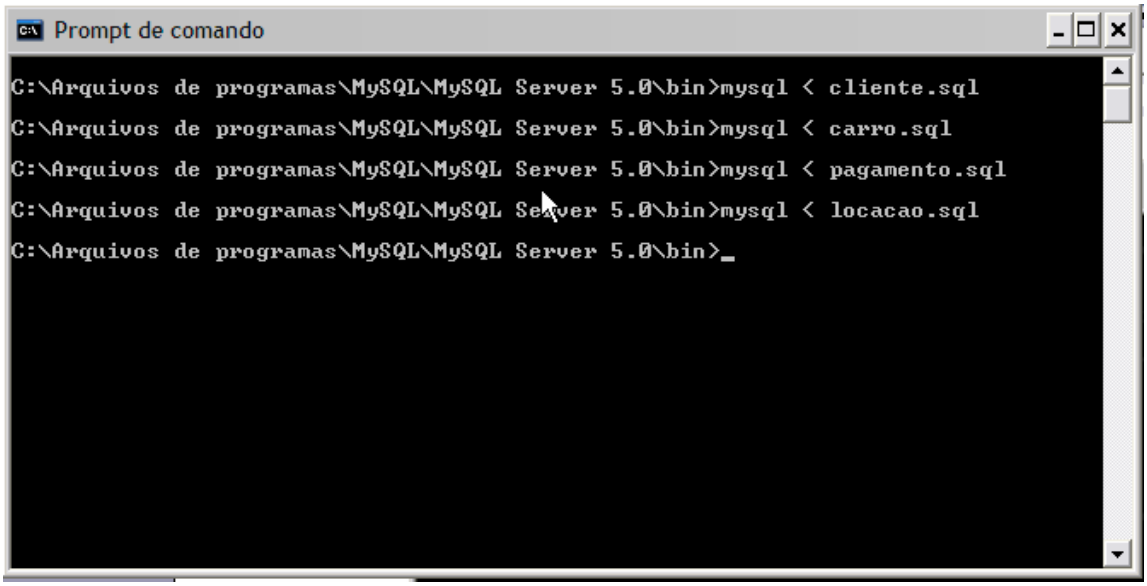

Figura C7 – Execução dos scripts no Mysql

Após a execução dos passos apresentados anteriormente, as tabelas estarão criadas na base de dados do SGBD Mysql.

ANEXO D – ERRO *swing-layout-1.0.jar*

Neste anexo (Figura D1) é apresentado o erro quando a API (*Application Programming Interface*) `Swing Layout` não está configurada.

```
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 173    1194095349515     28
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 176    1194095349515     29
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 178    1194095349515     30
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 179    1194095349515     31
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 182    1194095349515     32
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 184    1194095349515     33
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 186    1194095349515     34
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 188    1194095349515     35
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 190    1194095349515     36
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 192    1194095349515     37
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 195    1194095349515     38
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 196    1194095349515     39
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 196    1194095349515     40
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 200    1194095349515     41
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 202    1194095349515     42
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 210    1194095349515     43
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 211    1194095349515     44
org.jdesktop cannot be resolved     locadora/src/model      Maintenance.java
      line 212    1194095349515     45
org.jdesktop cannot be resolved to a type locadora/src/model
      Maintenance.java  line 170    1194095349515     26
      org.jdesktop cannot be resolved to a type locadora/src/model
      Maintenance.java  line 170    1194095349515     27
```

Figura D1 – Erro quando a API `Swing Layout` não está configurada

ANEXO E – ERRO *RESULTSET*

Neste anexo (Figura E1) é apresentado o erro quando um objeto da classe ResultSet é encerrado no método `DomainDB()` da classe `ResultSet()`.

```
ResultSet is closed
java.sql.SQLException: ResultSet is closed
	at sun.jdbc.odbc.JdbcOdbcResultSet.checkOpen(Unknown Source)
	at sun.jdbc.odbc.JdbcOdbcResultSet.clearWarnings(Unknown Source)
	at sun.jdbc.odbc.JdbcOdbcResultSet.close(Unknown Source)
	at model.DomainDB.getOid(DomainDB.java:372)
	at model.Ctrlmaintenance.incluir(Ctrlmaintenance.java:39)
	at model.Maintenance.adicionarActionPerformed(Maintenance.java:276)
	at model.Maintenance.access$1(Maintenance.java:260)
	at model.Maintenance$2.actionPerformed(Maintenance.java:81)
	at javax.swing.AbstractButton.fireActionPerformed(Unknown Source)
	at javax.swing.AbstractButton$Handler.actionPerformed(Unknown Source)
	at javax.swing.DefaultButtonModel.fireActionPerformed(Unknown Source)
	at javax.swing.DefaultButtonModel.setPressed(Unknown Source)
	at javax.swing.plaf.basic.BasicButtonListener.mouseReleased(Unknown
Source)
	at java.awt.Component.processMouseEvent(Unknown Source)
	at javax.swing.JComponent.processMouseEvent(Unknown Source)
	at java.awt.Component.processEvent(Unknown Source)
	at java.awt.Container.processEvent(Unknown Source)
	at java.awt.Component.dispatchEventImpl(Unknown Source)
	at java.awt.Container.dispatchEventImpl(Unknown Source)
	at java.awt.Component.dispatchEvent(Unknown Source)
	at java.awt.LightweightDispatcher.retargetMouseEvent(Unknown Source)
	at java.awt.LightweightDispatcher.processMouseEvent(Unknown Source)
	at java.awt.LightweightDispatcher.dispatchEvent(Unknown Source)
	at java.awt.Container.dispatchEventImpl(Unknown Source)
	at java.awt.Window.dispatchEventImpl(Unknown Source)
	at java.awt.Component.dispatchEvent(Unknown Source)
	at java.awt.EventQueue.dispatchEvent(Unknown Source)
	at java.awt.EventDispatchThread.pumpOneEventForFilters(Unknown
Source)
	at java.awt.EventDispatchThread.pumpEventsForFilter(Unknown Source)
	at java.awt.EventDispatchThread.pumpEventsForHierarchy(Unknown
Source)
	at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
	at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
	at java.awt.EventDispatchThread.run(Unknown Source)
java.sql.SQLException: ResultSet is closed
	at sun.jdbc.odbc.JdbcOdbcResultSet.checkOpen(Unknown Source)
	at sun.jdbc.odbc.JdbcOdbcResultSet.clearWarnings(Unknown Source)
	at sun.jdbc.odbc.JdbcOdbcResultSet.close(Unknown Source)
	at model.DomainDB.getOid(DomainDB.java:372)
	at model.Ctrlmaintenance.incluir(Ctrlmaintenance.java:40)
	at model.Maintenance.adicionarActionPerformed(Maintenance.java:276)
	at model.Maintenance.access$1(Maintenance.java:260)
	at model.Maintenance$2.actionPerformed(Maintenance.java:81)
	at javax.swing.AbstractButton.fireActionPerformed(Unknown Source)
	at javax.swing.AbstractButton$Handler.actionPerformed(Unknown Source)
	at javax.swing.DefaultButtonModel.fireActionPerformed(Unknown Source)
```

```
      at javax.swing.DefaultButtonModel.setPressed(Unknown Source)
      at javax.swing.plaf.basic.BasicButtonListener.mouseReleased(Unknown
Source)
      at java.awt.Component.processMouseEvent(Unknown Source)
      at javax.swing.JComponent.processMouseEvent(Unknown Source)
      at java.awt.Component.processEvent(Unknown Source)
      at java.awt.Container.processEvent(Unknown Source)
      at java.awt.Component.dispatchEventImpl(Unknown Source)
      at java.awt.Container.dispatchEventImpl(Unknown Source)
      at java.awt.Component.dispatchEvent(Unknown Source)
      at java.awt.LightweightDispatcher.retargetMouseEvent(Unknown Source)
      at java.awt.LightweightDispatcher.processMouseEvent(Unknown Source)
      at java.awt.LightweightDispatcher.dispatchEvent(Unknown Source)
      at java.awt.Container.dispatchEventImpl(Unknown Source)
      at java.awt.Window.dispatchEventImpl(Unknown Source)
      at java.awt.Component.dispatchEvent(Unknown Source)
      at java.awt.EventQueue.dispatchEvent(Unknown Source)
      at java.awt.EventDispatchThread.pumpOneEventForFilters(Unknown
Source)
      at java.awt.EventDispatchThread.pumpEventsForFilter(Unknown Source)
      at java.awt.EventDispatchThread.pumpEventsForHierarchy(Unknown
Source)
      at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
      at java.awt.EventDispatchThread.pumpEvents(Unknown Source)
      at java.awt.EventDispatchThread.run(Unknown Source)
INSERT INTO tbCliente(oid,codigo,nome) VALUES('0','0','')
```

Figura E1 – Erro `ResultSet`