deu ce

CENTRO UNIVERSITÁRIO DE BRASÍLIA -UniCEUB

CURSO DE ENGENHARIA DE COMPUTAÇÃO

Vitor Tormin Nishi

Middleware Acadêmico de Integração Microcontrolador e Banco de Dados

Orientador: Prof. Maria Marony Sousa Farias

Brasília

Novembro, 2012

Vitor Tormin Nishi

Middleware Acadêmico de Integração Microcontrolador e Banco de Dados

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Prof.

Maria Marony Sousa Farias

Brasília

Novembro, 2012

Vitor Tormin Nishi

Middleware Acadêmico de Integração Microcontrolador e Banco de Dados

Trabalho apresentado ao Centro Universitário de Brasília (UniCEUB) como pré-requisito para a obtenção de Certificado de Conclusão de Curso de Engenharia de Computação.

Orientador: Prof.

Maria Marony Sousa Farias

Este Trabalho foi julgado adequado para a obtenção do Título de Engenheiro de Computação, e aprovado em sua forma final pela Faculdade de Tecnologia e Ciências Sociais Aplicadas -FATECS.

_____

Prof. Abiezer Amarilia Fernandes
Coordenador do Curso

Banca Examinadora:

_____

Prof. nome, titulação.
Orientador

_____

Prof. nome, titulação.
Instituição

_____

Prof. nome, titulação.
Instituição

_____

Prof. nome, titulação.
Instituição

AGRADECIMENTOS

Aos meus pais, que me apoiaram e deram todo suporte necessário durante minha jornada acadêmica.

Aos professores e orientadores Javier e Marony, pelo suporte e paciência

Ao coordenador do curso Abiezer , pela paciência, principalmente.

A todos os professores do curso de Engenharia da Computação, que estão ou passaram pelo UniCEUB.

Aos meu amigos que me ajudaram neste passo importante, em destaque para o senhor Alderico Queiroz Junior e minha companheira Tais Moura Xavier.

SUMÁRIO

LISTA DE FIGURAS.....VI
LISTA DE SIGLAS.....X
RESUMO.....XI
ABSTRACT.....XII

1 - INTRODUÇÃO..... 1
2- APRESENTAÇÃO DO PROBLEMA..... 4
3 - BASES METODOLÓGICAS PARA RESOLUÇÃO DO PROBLEMA..... 7
4 - MODELO PROPOSTO..... 10
4.1 – Apresentação Geral do Modelo Proposto..... 10
4.2 – Descrição das Etapas do Modelo..... 11
4.3 - Descrição da implementação..... 12
5 - APLICAÇÃO DO MODELO PROPOSTO..... 43
5.1 - Apresentação do área de Aplicação do modelo..... 43
5.2 – Descrição da Aplicação do Modelo..... 44
5.3 – Avaliação Global do Modelo..... 52
6 - CONCLUSÕES..... 56
6.1 - Conclusões..... 56
6.2 - Sugestões para Trabalhos Futuros..... 57
REFERÊNCIAS 58

APÊNDICE A..... 59
APÊNDICE B.....87
APÊNDICE C..... 97
APÊNDICE D..... 106
APÊNDICE E..... 116
APÊNDICE F..... 124
APÊNDICE G..... 135

LISTA DE FIGURAS

Figura 2.1 - Arquitetura Comercial 4
Figura 2.2 - Arquitetura Acadêmica 5
Figura 2.3 - Problema 6
Figura 4.1 – Modelo 10
Figura 4.2 - Visão Geral 12
Figura 4.3 - Regras de Negócio 14
Figura 4.4 - Interfaces 15
Figura 4.5 - Parametrização 16
Figura 4.6 - Log 16
Figura 4.7 - Tabela de Requisitos 17
Figura 4.8 - Diagrama principal 18
Figura 4.9 - Modelo de Dado 19
Figura 4.10 - Diagrama de validação de Dados 20
Figura 4.11 - Diagrama de definição de operação 21
Figura 4.12 - Diagrama da Entrada de Dados Registro 22
Figura 4.13 - Leitura Serial 23
Figura 4.14 - Escrita Serial 24
Figura 4.15 - Leitura USB 25
Figura 4.16 - Diagrama banco de dados 27
Figura 4.17 - Diagrama de busca de parâmetros 29
Figura 4.18 - Diagrama Log 30
Figura 4.19 - Tabela de Pré-Requisitos 31
Figura 4.20 - Tabela de Rastreabilidade 32
Figura 4.21 - Package Main 33
Figura 4.22 - Classe Rotina 33
Figura 4.23 - Package Serial 35
Figura 4.24 - Classe SerialCom 35
Figura 4.25 - Classe SerialComLeitura 36
Figura 4.26 - Classe SerialConnector 37
Figura 4.27 - Package BancodeDados 38
Figura 4.28 - Classe BancoOperacoes 38
Figura 4.29 - Classe GetProperties 39

Figura 4.30 - Package Loader 40
Figura 4.31 - Package Log 41
Figura 4.32 - Classe ConfigLog 41
Figura 5.1 - Padrão Proposto 43
Figura 5.2 – DLL 44
Figura 5.3 – Lib 45
Figura 5.4 – Exe 46
Figura 5.5 – Carga 46
Figura 5.6 – Modo 47
Figura 5.7 – Entrada Dados 47
Figura 5.8 – Recebimento 48
Figura 5.9 – Operação 49
Figura 5.10 – Insert 50
Figura 5.11 – Registro Adicional 50
Figura 5.12 – Registro 51
Figura 5.13 – Consulta 51
Figura 5.14 – DesejaContinuar 52
Figura 6.1 - Conclusões 57
Figura B.1 - Arquivo parametrizado 87
Figura B.2 - Validação 88
Figura B.3 - Inicio Carga 88
Figura B.4 - Modo Entrada 89
Figura B.5 - Modo Serial 89
Figura B.6 - Aguardando dados 90
Figura B.7 - Enviar Dados 90
Figura B.8 - Dados Recebidos 91
Figura B.9 - Dado Válido 91
Figura B.10 - Operação Insert 92
Figura B.11 - Final Operação 92
Figura B.12 - Pesquisar 93
Figura C.1 - Arquivo parametrizado 97
Figura C.2 - Validação 98
Figura C.3 - Inicio Carga 98
Figura C.4 - Modo Entrada 99

Figura C.5 - Modo USB 99
Figura C.6 - Aguardando dados 100
Figura C.7 - Enviar Dados 100
Figura C.8 - Dados Recebidos 101
Figura C.9 - Dado Válido 101
Figura C.10 - Operação Insert 102
Figura C.11 - Final Operação 102
Figura C.12 - Pesquisar 103
Figura D.1 - Arquivo parametrizado 106
Figura D.2 - Validação 107
Figura D.3 - Inicio Carga 107
Figura D.4 - Modo Entrada 108
Figura D.5 - Modo Serial 108
Figura D.6 - Aguardando dados 109
Figura D.7 - Enviar Dados 109
Figura D.8 - Dados Recebidos 110
Figura D.9 - Dado Válido 110
Figura D.10 - Operação Registro 111
Figura D.11 - Dado Adicional 111
Figura D.12 - Final Operação 112
Figura D.13 - Pesquisar 112
Figura E.1 - Arquivo parametrizado 116
Figura E.2 - Validação 117
Figura E.3 - Inicio Carga 117
Figura E.4 - Modo Entrada 118
Figura E.5 - Modo USB 118
Figura E.6 - Aguardando dados 119
Figura E.7 - Dados Recebidos 119
Figura E.8 - Dado Válido 120
Figura E.9 - Operação Registro 120
Figura E.10 - Dado Adicional 121
Figura E.11 - Final Operação 121
Figura E.12 - Pesquisar 122
Figura F.1 - Arquivo parametrizado 124

Figura F.2 - Validação 125
Figura F.3 - Inicio Carga 125
Figura F.4 - Modo Entrada 126
Figura F.5 - Modo Serial 126
Figura F.6 - Aguardando dados 127
Figura F.7 - Enviar Dados 127
Figura F.8 - Dados Recebidos 128
Figura F.9 - Dado Válido 128
Figura F.10 - Operação Consulta 129
Figura F.11 - Dado Retornados 129
Figura F.12 - Final Operação 130
Figura F.13 - Retorno Serial 130
Figura G.1 - Arquivo parametrizado 135
Figura G.2 - Validação 136
Figura G.3 - Inicio Carga 136
Figura G.4 - Modo Entrada 137
Figura G.5 - Modo Serial 137
Figura G.6 - Aguardando dados 138
Figura G.7 - Enviar Dados 138
Figura G.8 - Dados Recebidos 139
Figura G.9 - Dado Válido 139
Figura G.10 - Operação Consulta 140
Figura G.11 - Dado Retornados 140
Figura G.12 - Final Operação 141
Figura G.13 - Retorno Serial 141

LISTA DE SIGLAS

RFID - *Radio-Frequency IDentification*

USB - *Universal Serial Bus*

API - *Application Programming Interface*

HID - *Human Interface Device*

DLL - *Dynamic-link library*

VSPE - *Virtual Serial Port Emulator*

SOA - *Service-Oriented Architecture*

RESUMO

Este trabalho apresenta um middleware voltado para integração de microcontroladores baseados em tecnologia RFID ou Código de Barras em um banco de dados relacional Oracle com três fluxos de dados definidos: Insert, Registro e Consulta.

Palavras Chave: Middleware, Integração, Microcontrolador, RFID e Código de barras.

ABSTRACT

This work shows a middleware pointed to an application integration with microcontrollers based on a RFID or Bar Code technology and a relational Oracle database with three well cleared flows: Insert, Registro e Consulta.

CAPÍTULO 1 - INTRODUÇÃO

1.1 – Apresentação do Problema

O Uso de tecnologias RFID, Código de barras ou outros sistemas baseados em Tags, etiquetas, labels é comum em projetos acadêmicos de conclusão dos cursos de engenharia, porém seu uso em geral é limitado ao hardware utilizado e a programação local do microcontrolador (quando utilizado).

Em aplicações comerciais o mesmo não ocorre, projetos como o descrito acima estão sempre integrados com sistemas de outras plataformas.

Esse abismo existente entre projetos acadêmicos e projetos comerciais ocorre, em parte, devido a necessidade de integração entre os sistemas. A integração entre sistemas demanda tempo de desenvolvimento e uma interdisciplinaridade que na maioria das vezes induz o projeto acadêmico a limitar-se ao hardware local, visto que é uma solução mais simples.

Dificuldades de integração não são apenas obstáculos no desenvolvimento de um projeto acadêmico, também são problemas que ocorrem em outras disciplinas de um curso de engenharia.

1.2 – Objetivos do Trabalho

O projeto tem como objetivo principal disponibilizar uma ferramenta de integração entre microcontroladores e bancos de dados.

Como objetivos secundários, podemos definir a implementação de três regras de negócios distintas, a implementação de um conjunto de interfaces com o microcontrolador e com o banco de dados, um sistema de portabilidade do software e um sistema de auditora.

1.3 – Justificativa e Importância do Trabalho

O middleware será a ferramenta que permitirá uma integração de forma simples entre microcontroladores e bancos de dados no meio acadêmico. Essa integração irá facilitar a interdisciplinaridade no curso de engenharia, principalmente entre as disciplinas de Microcontroladores, Arquitetura de Sistemas Distribuídos e Banco de

Dados e tornará possível o uso de exemplos ou desenvolvimento de projetos acadêmicos mais rebuscados, uma vez que tanto os exemplos de sala de aula, quantos os projetos terão uma facilidade na integração com outros sistemas.

1.4 – Escopo do Trabalho

O escopo do projeto é um software capaz de obter os dados na porta serial ou em uma interface HID USB, identificar a operação solicitada e aplicar a regra de negócio definida para cada operação.

O software realizará inicialmente uma carga de variáveis de ambiente, como dados de conexão com banco de dados, tabelas, colunas, porta serial e etc. de um arquivo de propriedades e posteriormente iniciará a leitura dos dados.

Os dados serão obtidos no formato de uma String, de qualquer tamanho (limitada ao tamanho da coluna onde ser inserida), acrescida do caráter ponto e virgula (‘;’) e outra String de três caracteres que deve ser uma das três operações definidas (INS, REG, CON), por exemplo “123456789;INS”, sendo que este formato será validado.

Após a leitura dos dados será definida a operação que irá encaminhar os dados para três fluxos de negócio distintos.

O primeiro fluxo irá inserir os dados em uma tabela, definida pelo usuário no arquivo de propriedades, com a data da leitura.

O segundo fluxo irá solicitar um dado adicional para o usuário e irá inserir o dado recebido do microcontrolador, o dado adicional e a data em uma tabela, também definida pelo usuário no arquivo de propriedades.

O terceiro fluxo irá consultar os dados recebidos no banco de dados e irá retornar o resultado de uma coluna, definida pelo usuário no arquivo de propriedades, para o microcontrolador na porta serial.

1.5 – Resultados Esperados

É esperado um software de fácil uso para desktops utilizando a plataforma Windows que integre projetos de microcontroladores com banco de dados para utilização em projetos de conclusão de curso e na disciplina de Microcontroladores, não apenas desta faculdade, mas de qualquer outra.

1.6 – Estrutura do Trabalho

No capítulo dois será apresentado de forma mais detalhada o problema identificado, deixando para o capítulo três introduzir conceitos importantes para o entendimento da solução, que será apresentada e no capítulo quatro, juntamente com os diagramas de caso de uso e de atividades, sendo que no capítulo cinco será apresentados os dados obtidos nos testes e por último, no capítulo seis será feita a conclusão.

CAPÍTULO 2 - APRESENTAÇÃO DO PROBLEMA

A flexibilidade, compactação, barateamento de tecnologias baseadas em RFID, código de barras e outros sistemas baseados em leitura de etiquetas vem expandindo a aplicação comercial de projetos nas mais diversas áreas, desde bibliotecas, aeroportos, hospitais, setores de tráfego, campanhas publicitárias e etc.

Essa expansão também vem crescendo em projetos acadêmicos, onde o interesse por este tipo de tecnologia vem se tornando frequente em projetos de graduação. Porém a abrangência do se uso é limitado em geral ao hardware, pois existe uma dificuldade em aplicar um arquitetura comercial, como exposto na figura 2.1 em um meio acadêmico.

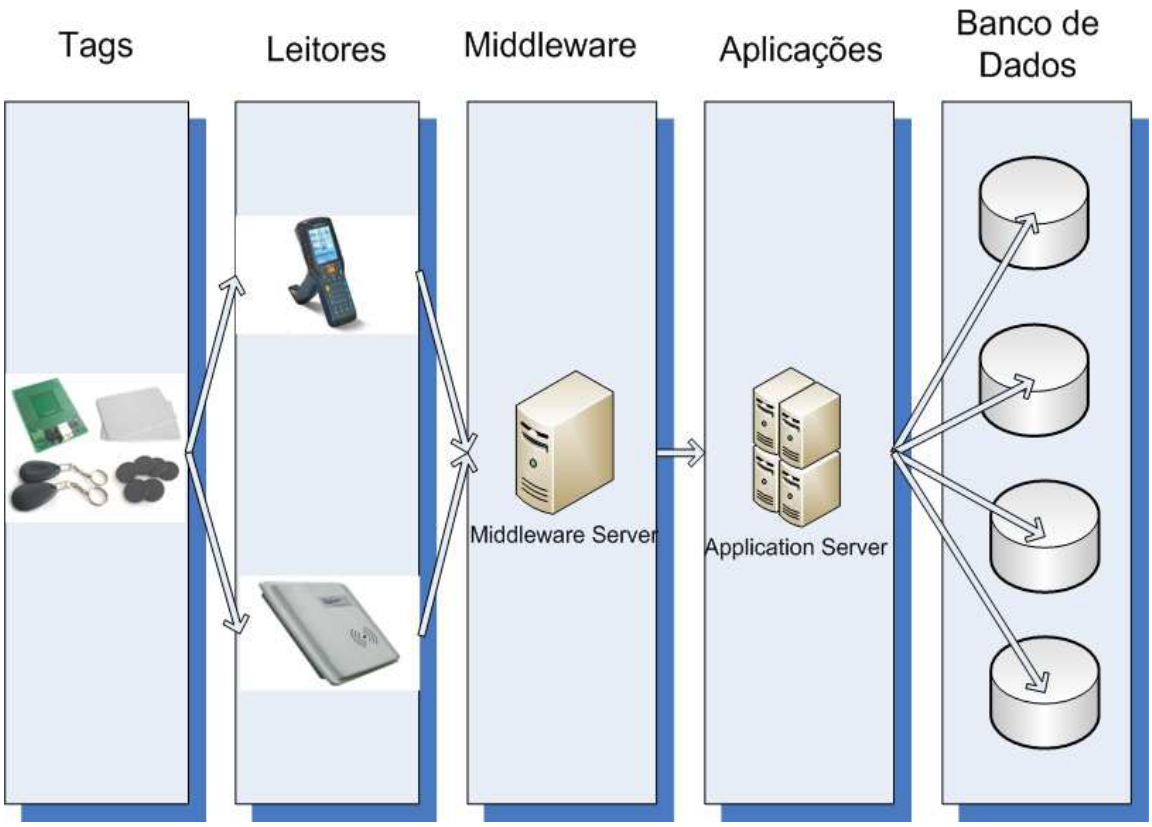

Figura 2. 1 - Arquitetura Comercial
Fonte: Autor

A integração entre o microcontrolador e outros sistemas de plataforma distribuída, em geral, é descartada na implementação do projeto acadêmico devido o tempo e dificuldade necessária para desenvolver um aplicativo para integrar as plataformas, uma vez que não existem aplicativos voltados para essa função, exceto por

aplicativos proprietários voltados para projetos comerciais que possuem um custo elevado, portanto os projetos acadêmicos acabam limitados à uma aplicação em hardware, como observado na figura 2.2.

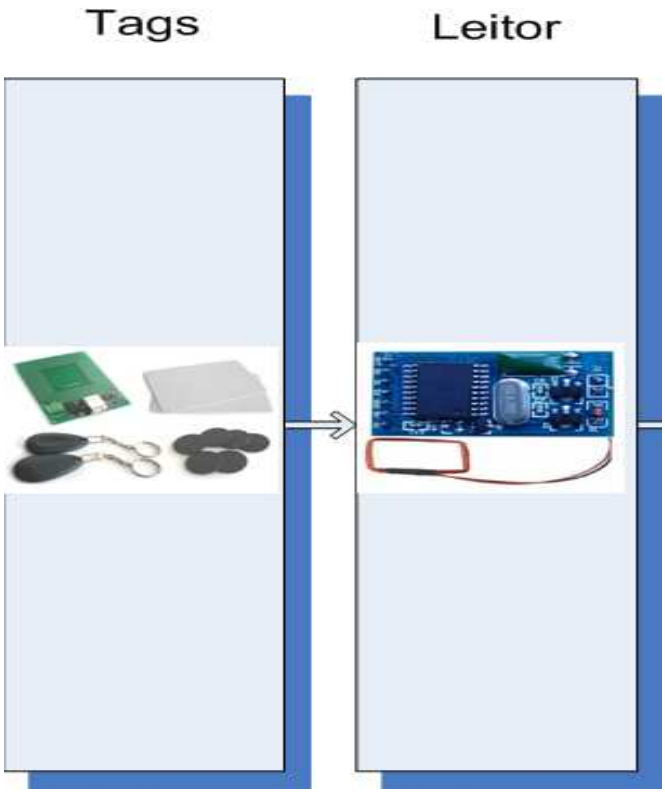

Figura 2. 2 – Arquitetura Acadêmica
Fonte: Autor

A limitação da integração não impacta apenas o desenvolvimento de projetos acadêmicos, ela também se torna uma limitação para que seja passada ao aluno uma visão integrada das disciplinas cursadas.

As disciplinas são cursadas, de maneira geral, isoladas uma das outras. Não é comum, por exemplo, em uma disciplina de Microcontroladores, ser exposto conteúdos de um disciplina como Banco de Dado; ou em uma disciplina de Redes sejam utilizados conhecimentos da disciplina de Engenharia de Software na construção de projetos; ou ainda na disciplina de Arquitetura de Sistemas Distribuídos para expor uma intranet.

O problema pode então ser agrupado em três classes, como visto na figura 2.3.

É compreensível que por limitações de tempo, muitas vezes tais aspectos devem ser deixados de lado no decorrer do curso, pois assim como no desenvolvimento de um projeto acadêmico, integrar as disciplinas requer tempo para desenvolvimento.

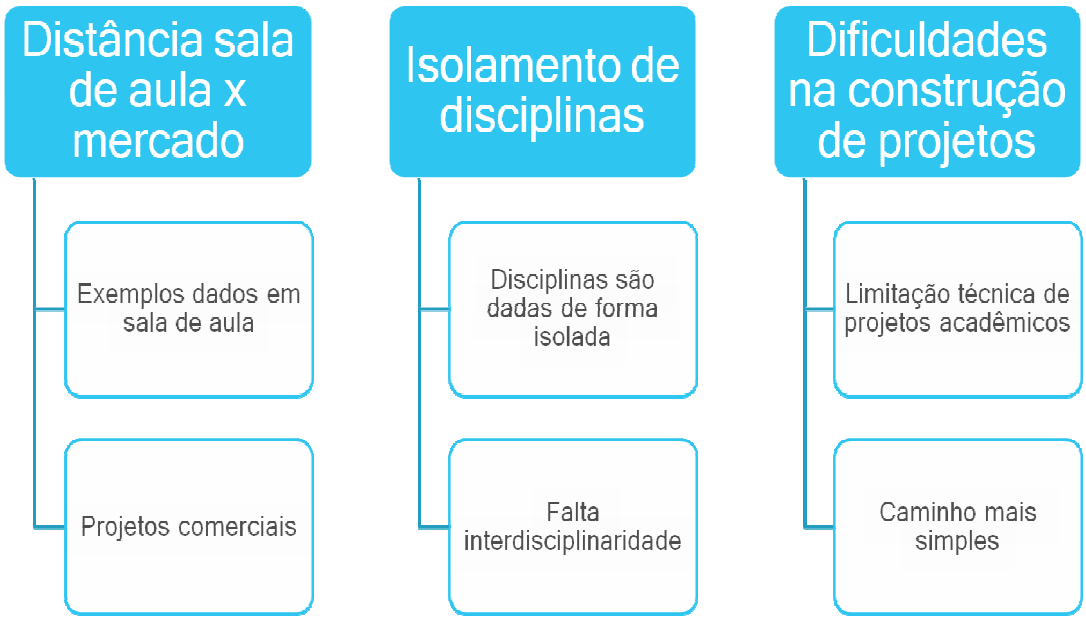

Figura 2. 3 – Problema
Fonte: Autor

CAPÍTULO 3 - BASES METODOLÓGICAS PARA RESOLULAÇÃO DO PROBLEMA

3.1 – Sistemas distribuídos

De acordo com o a definição: "*sistema distribuído é aquele no qual os componentes localizados em computadores interligados em rede se comunicam e coordenam suas ações apenas passando mensagens, Sistemas Distribuído - Conceitos e Projetos, George Coulouris*"**,** é possível inferir que um sistema distribuído é aquele no qual duas ou mais entidades computacionais conseguem, através da troca de mensagens, se comunicarem e se coordenarem em um ambiente comum para prover uma funcionalidade e/ou serviço.

Quando extrapolamos a citação de "*computadores*" para entidade computacional, podemos expandir a idéia de um sistema distribuído não apenas para desktop, servidores e mainframes, mas também para hardware mais básicos, como microcontroladores. Da mesma forma que ao interpretamos como ambiente comum a citação de "*rede*", podemos ver que um sistema distribuído existe não apenas sobre uma intranet ou internet, mas também em arquitetura de conexão mais básicas como uma comunicação direta via porta serial ou porta USB.

A ideia de expandir o conceito de sistemas distribuídos para aplicações mais simples como microcontroladores e portas seriais é utilizar conceitos e ferramentas desenvolvidos para computação distribuída em uma esfera mais próxima do hardware.
Um dos conceitos que serão utilizados é o de Middleware. Uma definição para este conceito é: "*o termo middleware se aplica a uma camada de software que fornece uma abstração de programação, assim como o mascaramento da heterogeneidade das redes, do hardware, de sistemas operacionais e linguagens de programação subjacentes, Sistemas Distribuído - Conceitos e Projetos, George Coulouris*". Este conceito é interessante ao aplicarmos em uma arquitetura de um desktop e microcontrolador, uma vez que temos hardwares distintos, sistemas operacionais distintos e linguagens distintas com uma necessidade comum de comunicação e coordenação.

Então, dando continuidade a idéia plantada acima, podemos entender que um software capaz de prover um comunicação e coordenação de forma transparente entre um ou mais desktops e um ou mais microcontroladores, através de uma intranet, internet

e/ou comunicação local via porta serial e/ou USB pode ser considerado um middleware, e nossa arquitetura um sistema distribuído, dentro da perspectiva apresentada acima.

3.2 – Linguagem de Programação

Pode-se definir uma linguagem de programação como um conjunto de regras sintáticas e semânticas que permitem a um programador o gerenciamento e manipulação de dados através de um estrutura lógica, denominada algoritmo, que uma vez compilado do código fonte para a linguagem de máquina pode ser executado pelo computador.

A ideia por traz do desenvolvimento e escolha de um linguagem de programação é permitir ao programador uma liberdade e facilidade ao construir um software, uma vez que este não precisará escrever em linguagem de máquina (código binário) e depender menos da arquitetura do hardware escolhido.

Para que seja possível escolher uma linguagem de programação que atenda a necessidade do programador, deve se observar qual o intuito e a arquitetura do projeto para o qual o software irá atender os requisitos pressupostos.

Considerando o intuito e a arquitetura descrita no tópico anterior, temos como requisito um software capaz de prover a comunicação entre um ou mais microcontroladores e um ou mais desktops, tanto através de uma comunicação local, quanto em rede. O software deverá além de prestar o serviço de mensageira dos sistemas também deverá orquestrar os mesmos.

Tendo em vista os requisitos acima, a escolha de uma linguagem adequada deve levar em consideração algumas características, como:

- Recursos disponíveis: Prover comunicação e gerenciamento de sistemas diversos requer uma linguagem com um leque vasto de bibliotecas e funcionalidades para atender diferentes tipos de comunicação, gerenciamento de dados e compatibilidade com hardwares distintos;
- Ser intuitiva: Além da linguagem ter os recursos, sua programação deve ser intuitiva para o programador. Da mesma forma que para indivíduo que fala naturalmente português aprender o inglês ou espanhol é me tese mais simples do que russo ou chinês, assim também deve ser a linguagem de programação escolhida;

Levando as características acima em conta, pode-se assumir que popularmente a escolha seria entre Java, C e C++, pois as três linguagens possuem uma quantidade enorme de bibliotecas, classes, procedures, funções, exemplos de código e todas são bastante intuitivas na construção de códigos, não é a toa que são as linguagens de programação mais populares.

Além das características citadas acima, a escolha pessoal do programador também deve ser levada em conta, assim como quando tomamos um refrigerante ou compramos um carro, sempre temos uma preferência pessoal.

Considerando a escolha pessoal, temos que a linguagem de programação escolhida para desenvolver o software para o sistema descrito anteriormente é Java.

O Java é uma linguagem orientada ao objeto que diferentemente de outras linguagens não compila seu código diretamente para linguagem de máquina, ela gera um código próprio, chamado de *bytecode* que é interpretado por uma máquina virtual Java. Isto permite uma maior portabilidade do software gerado para diversas plataformas de hardware e diversos sistemas operacionais.

A linguagem possui também outras características que são propícias para o projeto como um conjunto vasto de bibliotecas para comunicação em rede, comunicação com banco de dados, comunicação com as interfaces nativas dos sistemas operacionais para comunicação com hardware e outros (APIs) e facilidade para programação multitarefa ou mais conhecida como multitread.

CAPÍTULO 4 – MODELO PROPOSTO

4.1– Apresentação geral do modelo proposto

Neste capítulo é apresentado, discutido e aplicado o modelo de desenvolvimento escolhido para solução do problema apresentado no capítulo 2.

Como há um escopo definido do problema a ser tratado e há também um prazo definido para o desenvolvimento do projeto, ao considerar a simplicidade do mesmo e a flexibilidade para tratar com requisitos e pré-requisitos do projeto foi utilizado um modelo de desenvolvimento em cascata;, uma vez que o próprio autor é quem determina e levanta estes requisitos neste processo.

O modelo em cascata é definido com as seguintes etapas: **Requisitos**, **Solução**, **Implementação** e **Validação**, como observado na Figura 4.1.

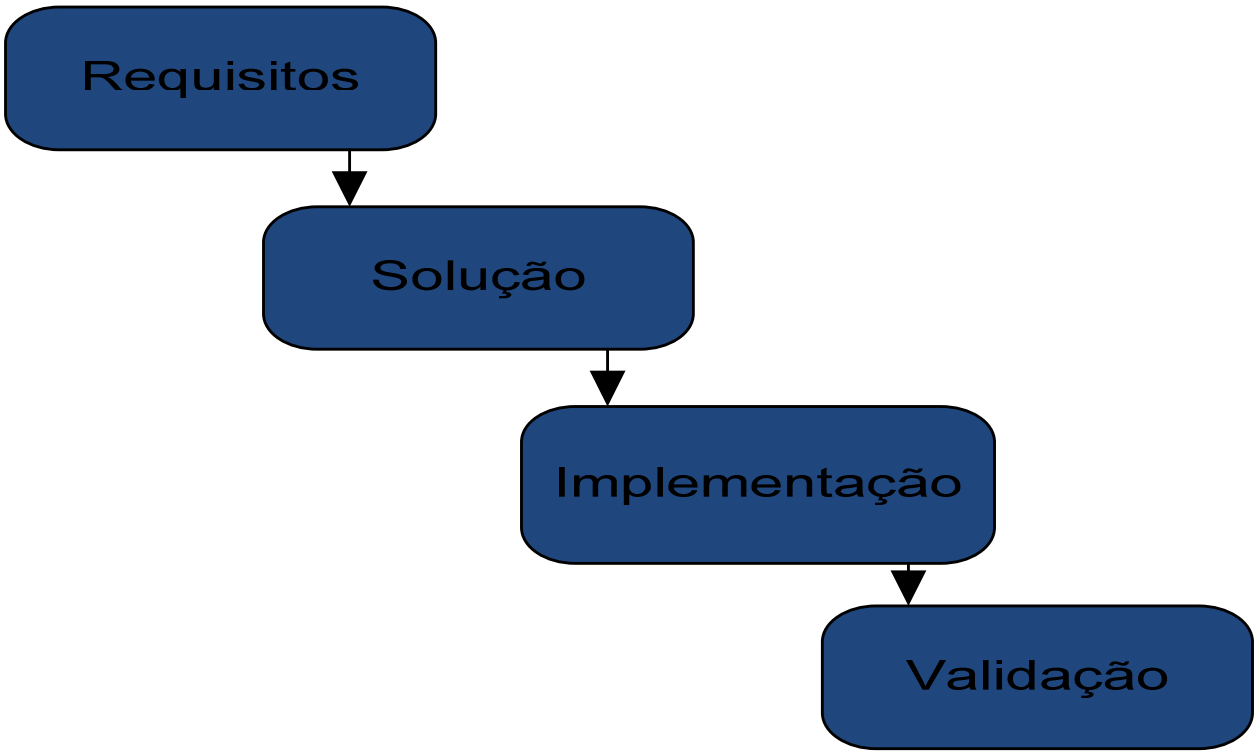

Figura 4. 1 – Modelo
Fonte: Autor

Na etapa de **Requisitos**, são levantados os requisitos funcionais do projeto. Para etapa de **Solução** é definido o algoritmo para solução de cada um dos requisitos definidos, bem como quais serão as premissas para atender cada um deles. Dentro da

etapa de **Implementação**, os algoritmos são implementados na linguagem Java. Por último, na etapa **Validação**, os recursos implementados são testados para garantir se os requisitos foram atendidos, dentro do conjunto de pré-requisitos proposto.

4.2– Descrição das etapas do modelo

A etapa **Requisitos**, como citado anteriormente, os requisitos funcionais do projeto são levantados e listados. Esta etapa é de fundamental importância, uma vez que todas as outras etapas dependem da tabela de requisitos que é gerada ao final desta etapa.

Nesta etapa, o autor analisa o problema proposto no capítulo 2 e define quais são as necessidades que o software a ser desenvolvido deve atender. Essas necessidades são listadas na forma de itens armazenados em tabela para facilitar a rastreabilidade de requisitos durante a execução das demais etapas e assim garantir ao final do projetos que todos os requisitos foram atendidos.

Ao final de etapa **Requisitos**, é iniciada a etapa **Solução**, onde são desenvolvidos os algoritmos, na forma de diagrama de atividades, para atender os requisitos levantados anteriormente, esses requisitos são analisados de forma individual ou em conjunto, quando necessário, para assim definir o algoritmo que atenda o demandado.

Durante esta etapa também são definidos o conjunto de premissas necessárias para que o algoritmo proposto atenda o requisito ou grupo de requisitos para o qual foi construído. Assim como os requisitos, os pré-requisitos também são agrupados na forma de tabela para facilitar a rastreabilidade.

Na etapa **Implementação** será implementado em código Java os algoritmos propostos na etapa anterior, que também utiliza os pré-requisitos propostos para definir os limites do código desenvolvido.

Ao final, na etapa **Validação**, o código desenvolvido é testado, validando se todos os itens da tabela de requisitos foram atendidos ou não e evidenciando o teste.

4.3– Descrição da implementação

Como descrito anteriormente, a primeira etapa é determinar os requisitos do projeto, mas para isso primeiro é necessário propor uma solução, para a partir dela identificar de forma mais apurada os requisitos do software.

Em um processo de desenvolvimento de software, o cliente ou usuário demanda um software no intuito de sanar um problema ou uma necessidade. Ao demandar o software, ele pode já informar como deseja que a solução seja implementada, propondo alguns requisitos primários ou pode deixar a cargo do responsável pelo levantamento de requisitos, que determine uma solução para a demanda proposta. No caso deste projeto, o usuário e o responsável por levantar os requisitos, são a mesma pessoa: o autor. Portanto, ambas as situações descritas acima se tornam o mesmo fluxo.

O problema que temos descrito no capítulo 2 é a integração de uma arquitetura de microcontroladores, baseados na leituras de tag e ou etiquetas, e um banco de dados. Mas o que seria essa integração? Uma visão geral da arquitetura pode ser visualisada na figura 4.2, porém para um melhor entendimento do que seria essa integração, temos de pensar em quais serviços e funcionalidades básicas uma arquitetura similar a listada anteriormente necessitam.

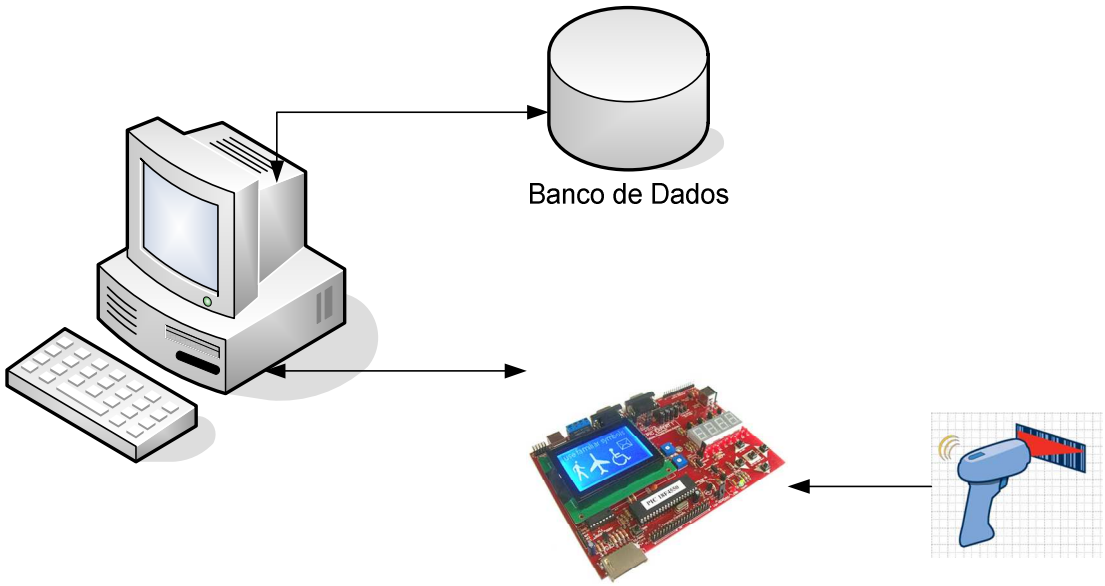

Figura 4. 2 – Visão Geral
Fonte: Autor

Iremos analisar primeiro uma arquitetura baseada em código de barras de um supermercado. Em um supermercado, ao passar um produto no leitor de códigos de barras, o preço do produto é exibido para o usuário, ou seja, o código de barras é lido, seu valor é utilizado como parâmetro de consulta em um banco de dados, onde o preço é consultado e retornado para o leitor que exibe o preço em um display.

O mesmo cenário pode ser encontrado em uma arquitetura de controle de acesso em edifícios, onde para acessar determinada localidade do prédio, o crachá com um tag RFID, será lido em um leitor RFID e pesquisado no banco de dados para verificar se aquela tag tem ou não acesso a localidade, autorização que será retornada para o leitor ou outro dispositivo que irá ou não abrir a porta, de acordo com o retorno. E diversos outros exemplos que tem esse cenário de consulta da tag ou etiqueta em um banco de dados, retornando o valor para o leitor. Portanto nosso software, deve disponibilizar um serviço de consulta.

Voltando ao exemplo do supermercado, antes de consultar o dado no banco, foi necessário primeiramente registrar aquele código de barras no banco e vinculá-lo ao produto. Processo semelhante ao que ocorre em bibliotecas, onde cada livro é registrado para uma tag RFID em banco de dados. Ou seja, outro serviço comum é registrar uma tag, etiqueta ou código de barras para um produto, pessoa ou qualquer outro objeto. Com isso o software deve disponibilizar um serviço de registro que permita após a leitura do código de barras ou tag, adicionar um novo dado, para inserir ambos dados no banco, vinculando ambos.

Existem cenários em que basta inserir a tag no banco, sem acréscimo de dados ou retorno para o leitor. Arquiteturas como do pedágio Sem Parar, basta que a identificar que a tag foi lida em determinado posto de pedágio, que a partir daí outros sistemas conseguem debitar o valor do pedágio de conta do usuário, mesma ideia pode ser aplicada em monitoramento de tráfego, para saber o trajeto do veículo, basta identificar que o código de barras, etiqueta ou tag foi lida em determinado leitor. Com isso temos que o software proposto também deve fornecer um serviço de insert, para apenas inserir o registro lido no banco.

Tem-se, então que a integração do software proposto se resume a 3 operações básicas: **Consulta**, **Registro** e **Insert**, como observado na figura 4.3.

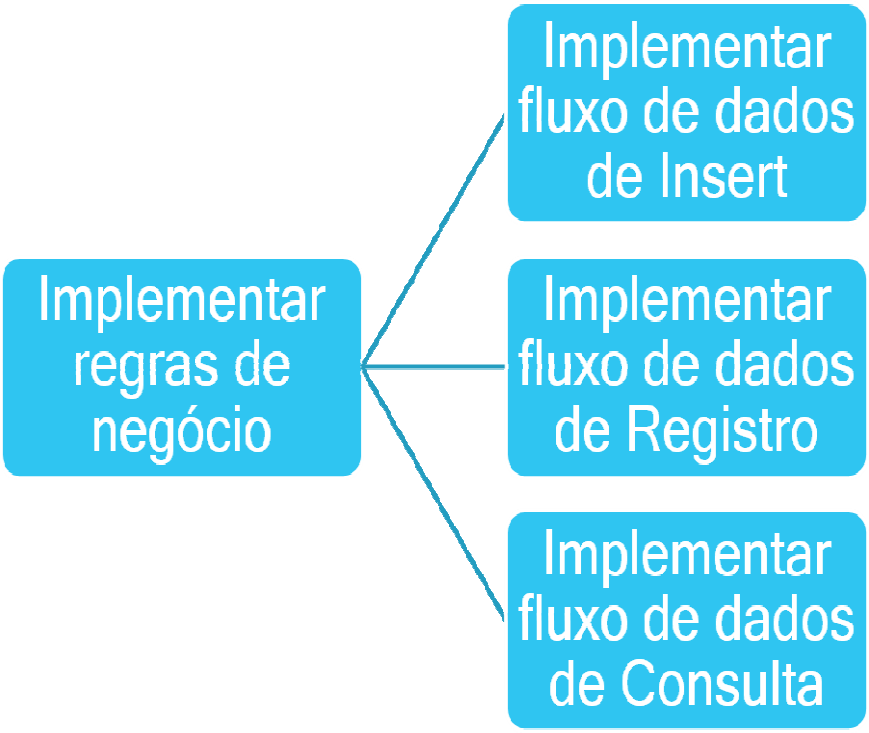

Figura 4. 3 – Regras de Negócio
Fonte: Autor

Mas, apenas isso não é suficiente, como citado anteriormente. O middleware deve adquirir os dados do microcontrolador e em alguns casos retornar dados para o mesmo, pois não é possível ter uma integração sem um serviço de leitura e escrita com alguma interface do microcontrolador. Para isso, deve-se pensar no hardware do microcontrolador, mais precisamente, quais portas de comunicação ele suporta. A família de processadores PIC adquiridos pelo UniCEUB no segundo semestre de 2012, possui suporte de comunicação serial apenas. Portanto, tendo em vista que os alunos de engenharia da computação do UniCEUB serão os usuários finais deste software, esta versão terá suporte para leitura e escrita de dados na porta serial.

Contudo, a interface de leitura não é sempre realizada pelo microcontrolador, em diversos projetos de conclusão do curso são utilizados leitores com comunicação na porta USB, ou seja a leitura não é feita pelo microcontrolador. Com isso, diferentemente do serviço escrita que será apenas na serial, o software aqui proposto também possui suporte para leitura de dados na porta USB. Então temos que o middleware deve possuir um serviço capaz de ler dados na porta serial, ler dados na porta USB e escrever dados na porta serial.

Os dados coletados tem que realizar operações com o banco de dados. Mas, primeiro, o software deve ser capaz de realizar uma conexão com o banco de dados,

então outro serviço no nosso middleware é que ele deve realizar conexões com bancos de dados. Uma vez conectado com o banco, teremos duas operações que o middleware irá realizar, uma é o **insert** de dados e outra o **select**. Com isso, temos que o aplicativo deve ser capaz de realizar uma ação de **insert** e outra de **select** no banco. Estes requisitos podem ser consolidados na figura 4.4.

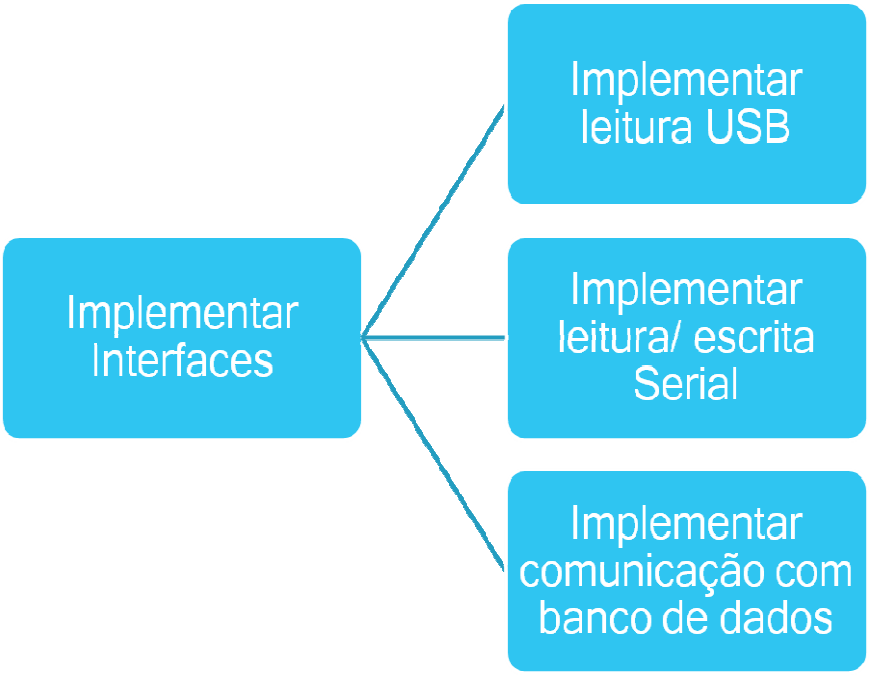

Figura 4. 4 – Interfaces
Fonte: Autor

Já foi citado que o software proposto deve possuir 3 operações básicas: **Registro**, **Consulta** e **Insert**, que ele deve ser capaz de ler e escrever dados na porta serial, também deve ler dados na porta USB, fazer conexão com um banco de dados, realizar uma operação de **insert** e outra de **select** no banco. Mas, o middleware não é voltado para uma arquitetura específica, como a de um supermercado ou biblioteca. Ele também deve ser portado e adaptável para vários modelos. Portanto este software deve ser parametrizável. Deve ser possível que o usuário parametrize o banco de dados para conexão, as tabelas e colunas envolvidas, bem como qual o meio de entrada de dados (Serial ou USB) e as características da porta serial. Então, outro serviço do middleware é uma interface de configuração de parâmetros de ambiente, como descrito na figura 4.5.

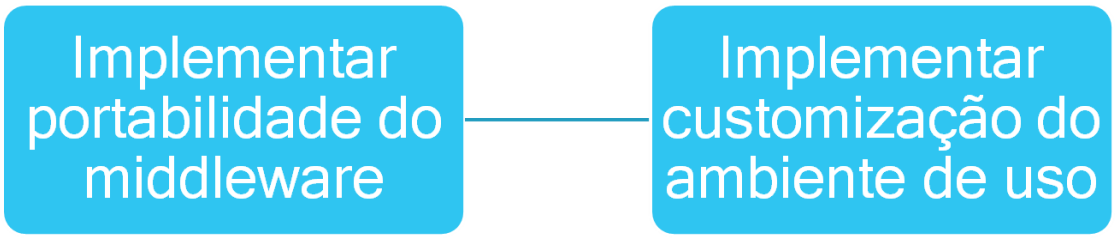

Figura 4. 5 – Parametrização
Fonte: Autor

As ações realizadas pelo middleware, por uma das características do software que é ser transparente ao usuário, podem tornar complicado avaliar se o requisito está sendo atendido ou não. Portanto, outro serviço que o middleware fornece é um meio de gravação de logs das atividades executadas, conforme figura 4.6.

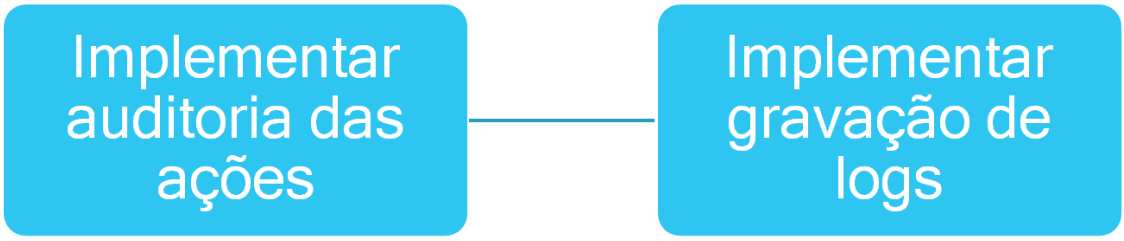

Figura 4. 6 – Log
Fonte: Autor

Com isso podem ser geradas tabelas dos requisitos do projeto, conforme pode ser observado na figura 4.7.

Req01	O software deve fornecer um fluxo para realizar um insert do dado lido na porta serial ou USB no banco de dados
Req02	O software deve fornecer um fluxo para realizar um registro do dado lido na porta serial ou USB e um dados adicional no banco de dados
Req03	O software deve fornecer um fluxo para realizar uma consulta do dados lido na porta serial ou USB em um banco de dados e retornar um valor para a porta serial
Req04	O software deve disponibilizar um serviço de leitura na porta serial
Req05	O software deve disponibilizar um serviço de leitura na porta USB
Req06	O software deve disponibilizar um serviço de escrita na porta serial
Req07	O software deve disponibilizar um serviço de conexão com banco de dados
Req08	O software deve disponibilizar um serviço de select no banco de dados
Req09	O software deve disponibilizar um serviço de insert no banco de dados
Req10	O software deve disponibilizar um serviço de parametrização do banco de dados
Req11	O software deve disponibilizar um serviço de parametrização das tabelas do banco de dados
Req12	O software deve disponibilizar um serviço de parametrização das colunas do banco de dados
Req13	O software deve disponibilizar um serviço de parametrização do meio de entrada de dados (Serial ou USB)
Req14	O software deve disponibilizar um serviço de parametrização da porta serial que será realizada a leitura e/ou escrita
Req15	O software deve disponibilizar um serviço de parametrização do baudrate da porta serial
Req16	O software deve disponibilizar um serviço de parametrização do timeout de abertura da porta serial e de leitura da porta
Req17	O software deve disponibilizar um serviço log para as atividades realizadas

Figura 4. 7 – Tabela de Requisitos
Fonte: Autor

Com a etapa de **Requisitos** concluída, pode-se prosseguir para etapa **Soluções**, onde serão definidos os algoritmos, na forma de diagrama de atividades, para atender os requisitos propostos. Nessa etapa também são definidas os pré-requisitos para que os requisitos sejam atendidos.

Para atender os requisitos Req01, Req02 e Req03, temos que o corpo principal para que o programa siga o diagrama de atividades proposto na figura 4.8.

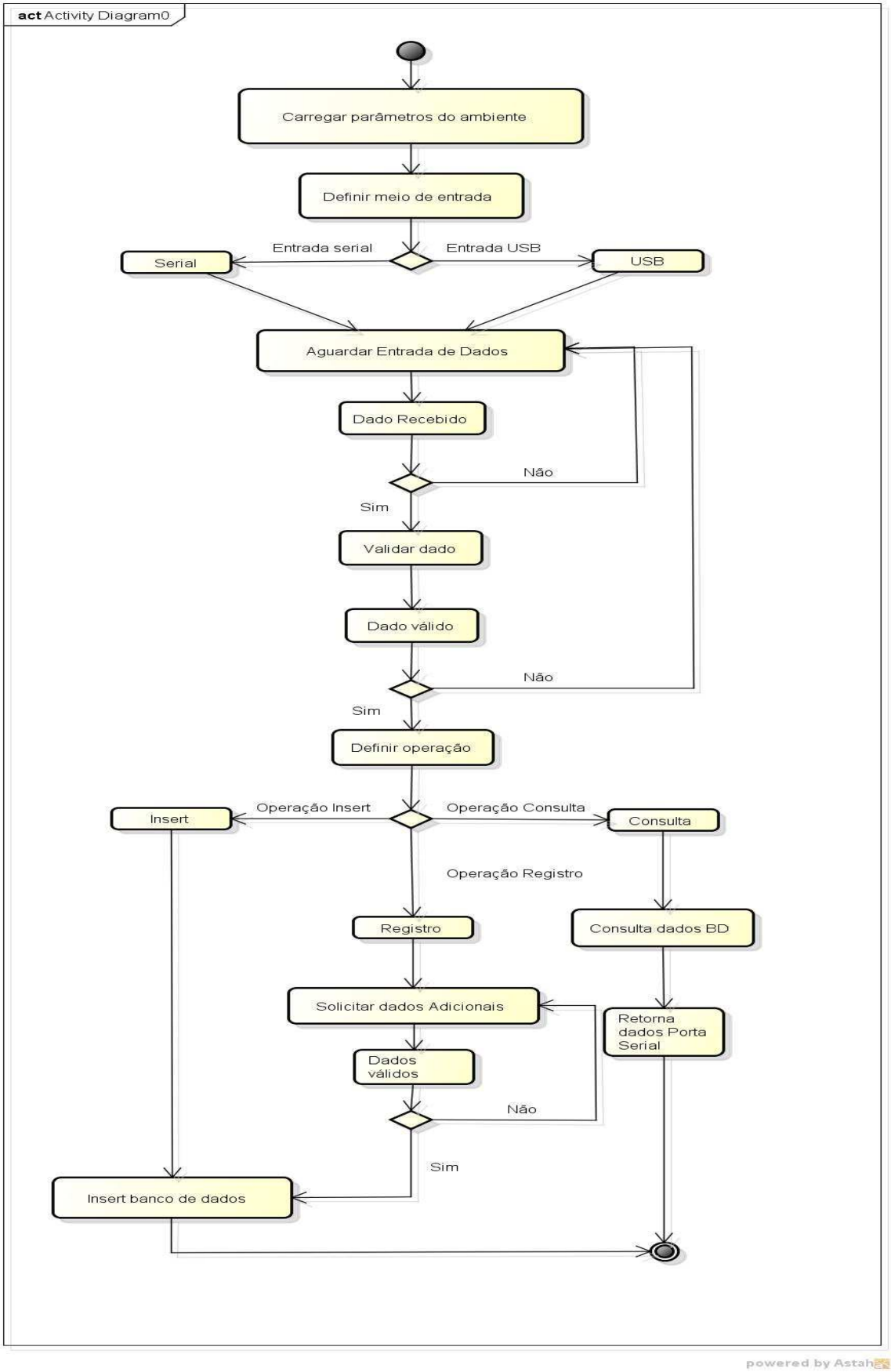

Figura 4. 8 – Diagrama Principal
Fonte: Autor

Dentro do diagrama de atividades da Figura 4.8 temos algumas validações e definições que o software executa. A primeira é a definição do meio de entrada, que é definido de acordo com o valor do parâmetro estabelecido pelo usuário, porém o detalhamento dessa função será tratado posteriormente, quando forem abordados os requisitos 10 a 16.

Na sequência, temos a definição se o dado é ou não válido, para isso termos nosso primeiro pré-requisito. Para ser considerado um dado válido, é estabelecido um padrão da forma que o dado deve ser enviado do microcontrolador ou leitora para o middleware. O padrão dos dados deve seguir o modelo da figura 4.9.

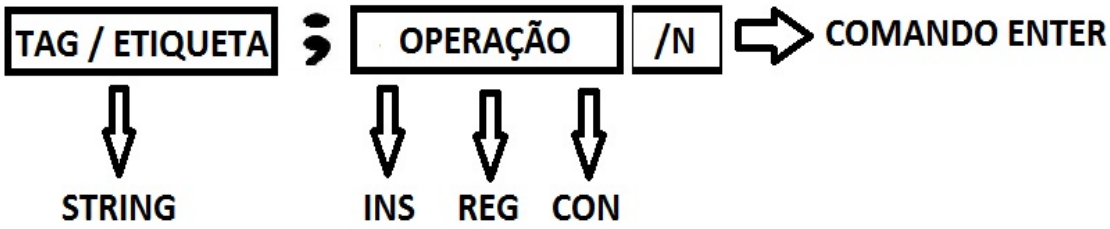

Figura 4. 9 – Modelo de Dado
Fonte: Autor

Com isso podemos definir nosso pré-requisito número 1: Preq01 – Todo dado enviado dever ser enviado na forma de uma string dividida em 4 partes, a primeira composta do valor da tag ou etiqueta, a segunda do caráter ponto e vírgula, a terceira de uma operação que deve ser INS, REG ou CON, de acordo com o fluxo de dados desejado e a quarta parte um comando /n para indicar um comando de “Enter” dos dados.

Ou seja o programa ficará em loop até que seja recebido um dado válido, conforme a figura 4.10.

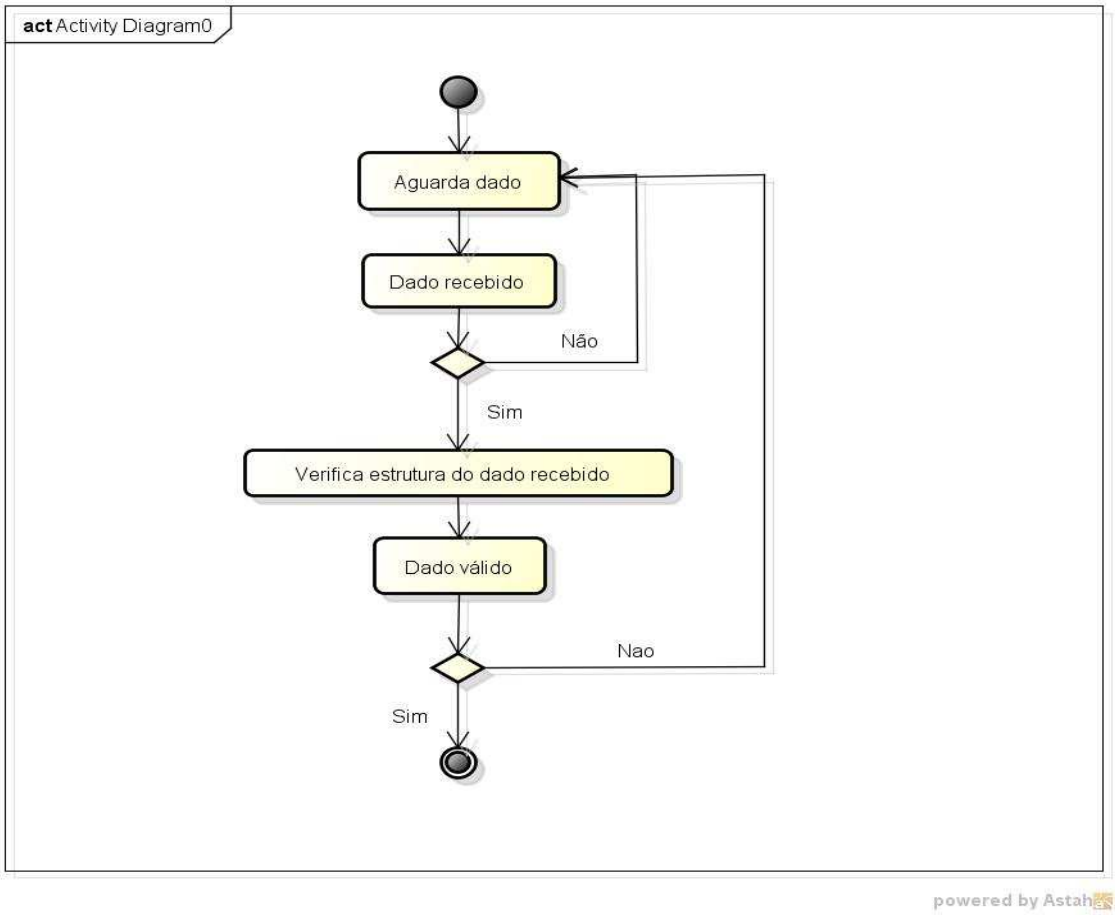

Figura 4. 10 – Diagrama de Validação de Dados
Fonte: Autor

Após a validação de estrutura, o programa define qual a operação enviada, de acordo com o terceiro trecho da string recebida, ou seja INS para Insert, REG para Registro e CON para Consulta, caso seja enviada uma string diferente de uma das 3 propostas, o programa informa ao usuário que a operação enviada não foi prevista e irá para fim, como observado na figura 4.11.

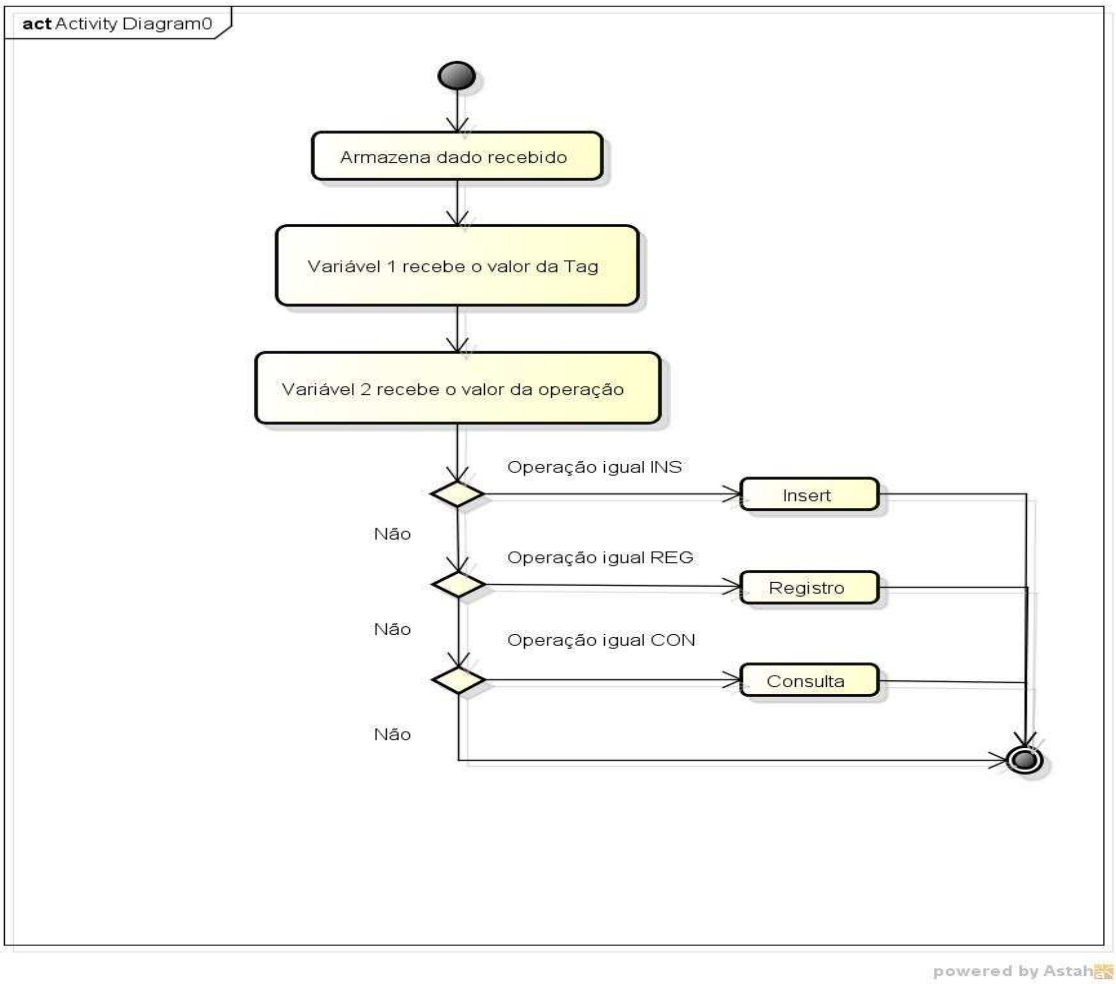

Figura 4. 11 – Diagrama de Definição de Operações
Fonte: Autor

A última validação realizada no programa principal é dentro da operação de registro. É solicitado ao usuário a entrada de um dado adicional para ser inserido no banco. Este dado não pode ser branco; portanto, o middleware entra dentro de um loop até que o usuário entre com um dado não nulo, diagrama da atividade pode ser visto na figura 4.12.

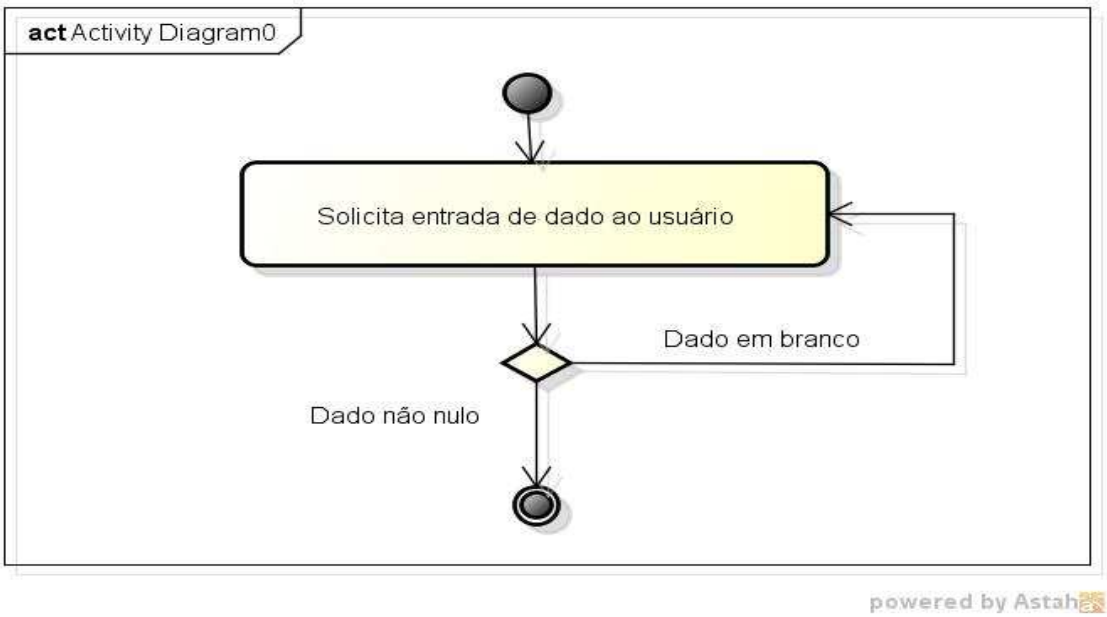

Figura 4. 12 – Diagrama de Entrada de Dados Registro
Fonte: Autor

Diagramas de nossa rotina principal foram definidos, agora vamos tratar dos requisitos Req04, Req05 e Req06.

Para leitura e escrita de dados na porta serial, deve-se definir novamente uma premissa, está premissa é: Preq02 – O nome da porta serial, o baudrate da porta serial e o timeout de operação devem ser parametrizados sempre pelo usuário, com valores não nulos.

Essa premissa deve ser determinada, pois para iniciarmos uma operação com a porta serial. Primeiro precisamos saber qual a porta serial a ser utilizada, daí a necessidade do nome. O *baudrate* é uma característica do hardware da porta e deve ser informado também para o ajuste da taxa de envio de dados e o tempo do **timeout** é necessário para determinar o tempo de espera da porta, tanto para abrir a porta, quanto para o tempo que a porta aguarda entrada de dados. O diagrama para leitura de dados pode ser observado na figura 4.13 e o para escrita na figura 4.14.

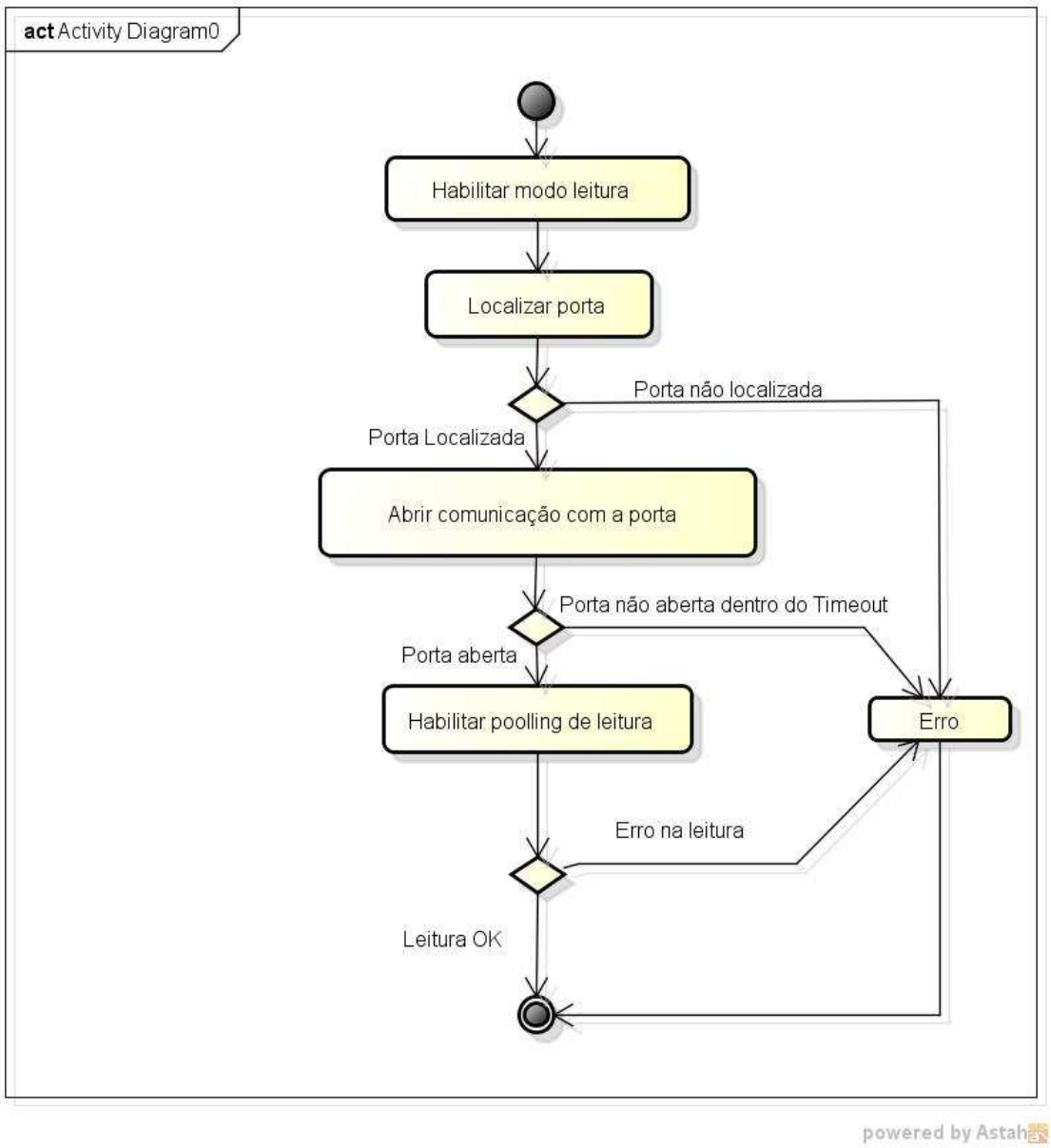

Figura 4. 13 – Leitura Serial
Fonte: Autor

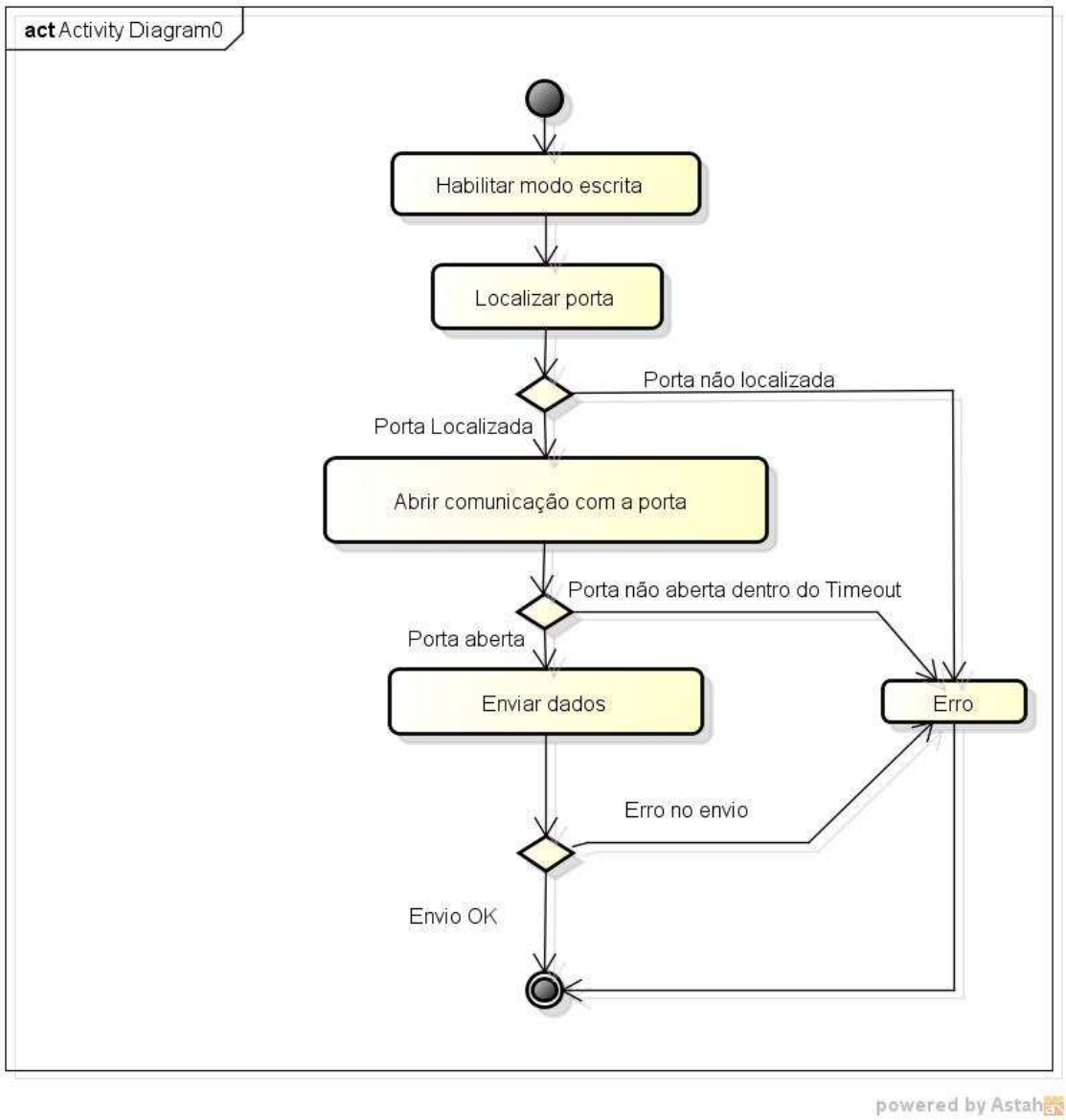

Figura 4. 14 – Escrita Serial
Fonte: Autor

Em ambos os processos é necessário um estágio para habilitar se o modo de operação da porta será Leitura ou Escrita, uma vez que a porta não pode operar das 2 maneiras ao mesmo tempo.

No modo de leitura temos que é habilitado um *poolling* de leitura, este *pooling* é necessário; pois, é o tempo que a porta serial é aberta aguardando dados, este tempo é configurado pelo usuário, conforme premissa Preq02 e seu tempo é o mesmo que o *timeou*t de tentativa de abertura da porta serial.

Para permitir a leitura da porta USB, teremos um novo pré-requisito, esta nova premissa é: Preq03 – O dispositivo USB, deve ser um dispositivo USB HID (*Human*

Interface Device), como teclados, mouses, leitores RFID e Leitores Código de barras comerciais.

Esta premissa é importante, pois a grande maioria dos dispositivos que utilizam a porta serial para entrada de dados como teclados, leitores e microcontroladores utilizam essa interface. Porém existem outras interface de entrada de dados via USB, como *streaming* de dados. Porém nossa aplicação não é voltada para esse tipo de interface, uma vez que não é comum hardwares de leitores ou microcontroladores que operem desta forma para as aplicações as quais se destinam o middleware aqui apresentado

Tendo como base a premissa Preq03, tem-se que a leitura do dados pela porta USB é simples, basta que o usuário envie o dado no formato da premissa Preq01. O modo de entrada USB é como a entrada de um dado no teclado, basta digitar o desejado e dar um Enter. Processo pode ser visto na figura 4.15.

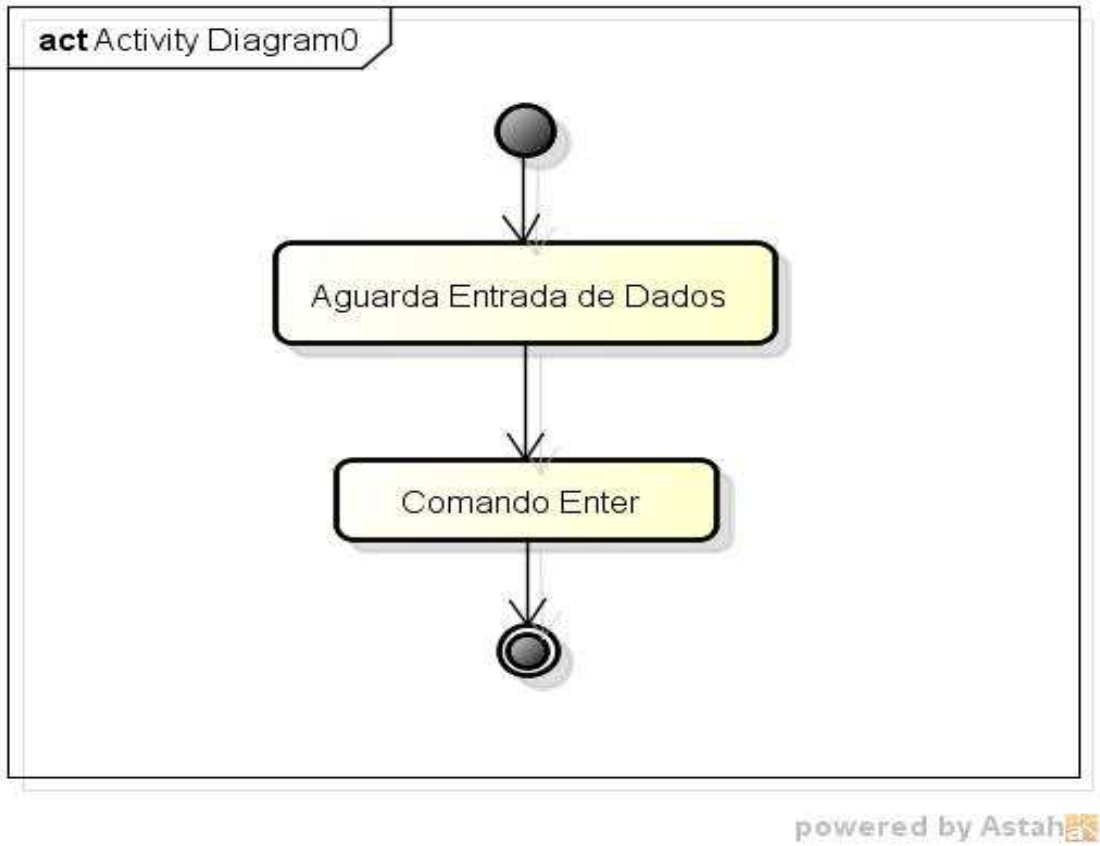

Figura 4. 15 – Leitura USB
Fonte: Autor

Agora temos os requisitos Req07, Req08 e Req09, novamente antes de determinar o algoritmo para solução, temos de determinar algumas premissas. Para

determinar estas premissas devemos identificar os elementos necessários para uma comunicação com banco de dados. Nosso primeiro elemento é o banco de dados. Presume-se que o usuário possua um banco de dados instalado localmente ou em sua intranet. Mas, apenas presumir que exista um banco não é suficiente, pois existem diversos fornecedores e modelos de bancos de dados. Portanto é necessário definir um banco e seu fornecedor. Para esta aplicação foi utilizado um banco de dados relacional da Oracle. A escolha do modelo e fornecedor é em virtude da facilidade de integração da linguagem Java e o banco de dados Oracle, umas vez que a linguagem Java pertence a Oracle. Com isso nosso novo pré-requisito é: Preq04 – Existência de um banco de dados Oracle local ou na intranet do usuário.

No entanto, somente a existência de um banco de dados não é suficiente para garantir a conexão, é necessário um usuário e senha válidos no banco. Portanto,uma nova premissa é: Preq05 – Existência de um usuário e senha válidos no banco.

Com essas duas premissas anteriores, pode-se garantir que o software cumprirá o requisito Req07. NO entanto, também temos os requisitos Req8 e Req9. Para esses requisitos, teremos que adicionar mais uma premissa. Esta nova premissa é: Preq06 – Todos os dados de tabela e coluna devem ser parametrizados pelo usuário previamente com objetos válidos no banco. Por último a premissa: Preq07 – O usuário do banco deve ter os acessos necessários aos objetos do banco para executar as ações de insert e select. Agora podemos definir o diagrama de funcionamento dos algoritmos de banco, como visto na figura 4.16.

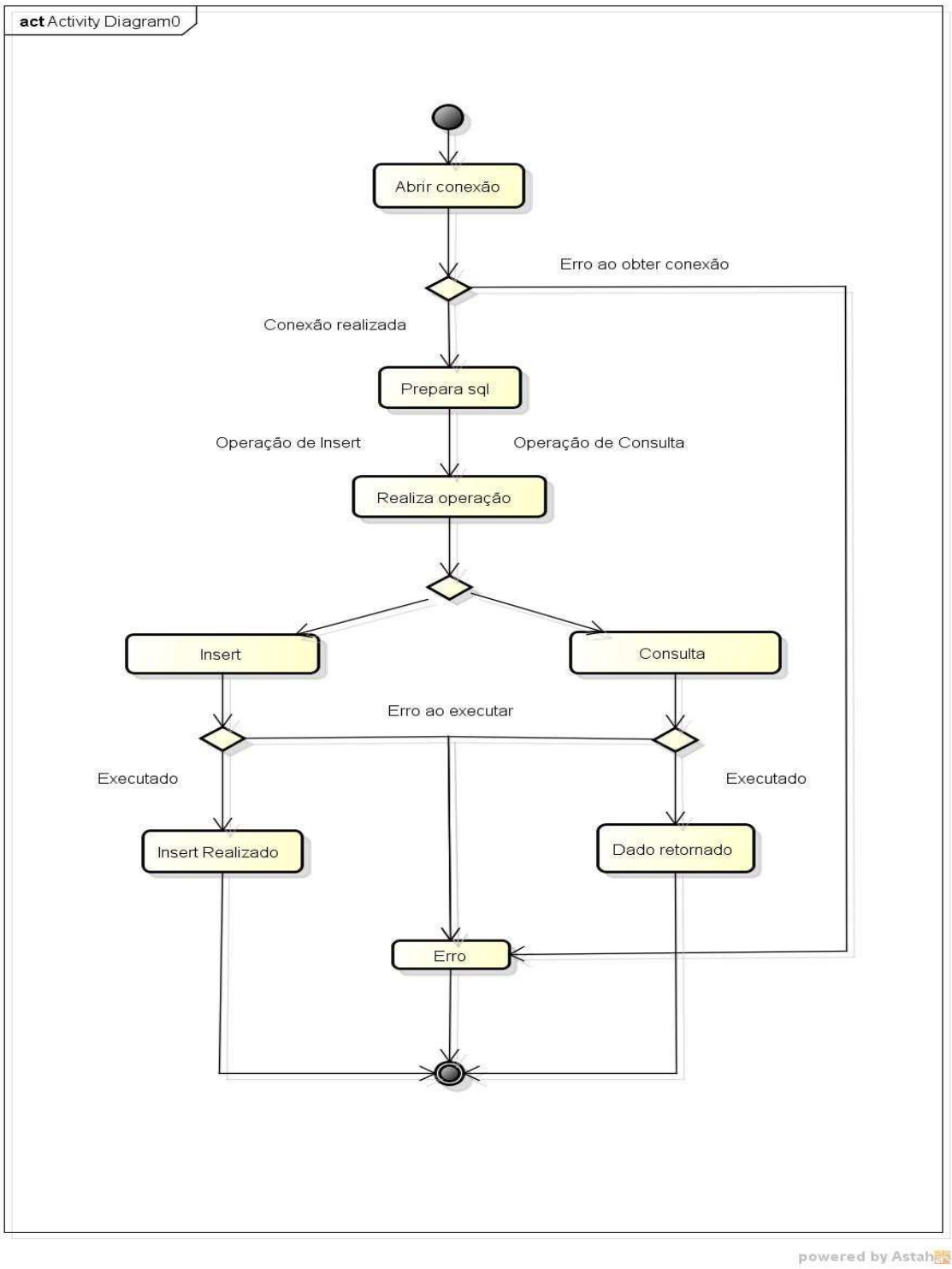

Figura 4. 16 – Diagrama Banco de Dados
Fonte: Autor

O algoritmo primeiro abre a conexão com o banco de dados. Caso a conexão não seja aberta ocorrerá um erro. Após a conexão aberta, é verificado qual a operação de banco deve ser realizada e o SQL de instrução é montado, nesse ponto os dados das

tabela e colunas parametrizadas pelo usuário são utilizados para montagem do comando SQL. Caso a instrução seja executada com erro, é retornando um erro.

Os requisitos do número 10 ao 16 tratam da parametrização do *middleware*. Deve ser apresentado agora uma solução de como realizar a parametrização. O software utiliza um arquivo *properties* para armazenar e buscar os parâmetros necessário. Deve-se então ter outras premissas, a primeira é: Preq08 – Deve existir um arquivo chamado "middleware.properties" dentro da pasta onde o executável do software for executado.

Outro pré-requisito que se tem é que todos os parâmetros do arquivo sejam devidamente preenchidos pelo usuário ou seja: Preq09 – Devem ser preenchidos corretamente todos os parâmetros do arquivo de configuração. Este arquivo deve ter um *layout* que é definido de seguinte forma:

#CONFIGURACAO DO NIVEL DE LOG
nivel_log = nível do log na rotina principal

# INFORMACOES DE CONEXAO COM A BASE DE DADOS
driver = Driver do fornecedor do banco de dados, no caso do projeto o valor é Oracle.jdbc.driver.OracleDriver
path = string de conexão com o banco de dados, deve conter o host, porta, usuário e senha do banco

# TABELA INSERT
tabelainsert = Tabela onde será inserido os dados da operação insert
colunainsert1 = Coluna onde o dado será inserido

# TABELA CONSULTA
tabelaconsulta = Tabela onde o dado será consultado
campo1tabela = Primeira coluna pesquisada, este dado será retornado para o microcontrolador
campo2tabela = Segunda coluna pesquisada, este dado não será retornado para o microcontrolador
campopesquisa = Coluna que o valor da tag será utilizado como parâmetro de consulta

TABELA REGISTRO

tabelaregistro = Tabela onde o insert dos dados da operação de registro será feito
colunaregistro1 = Coluna onde a Tag será inserida
colunaregistro2 = Coluna onde o dado adicional será inserido

# INFORMACAO DA PORTA
baudrate = Baudrate de porta
modoentrada = Modo de entrada da porta, SERIAL ou USB
portaserial = Nome da porta serial, COM1, COM2, COM3, etc.
sleep = Tempo do timeout em milissegundos

O diagrama de figura 4.17 mostra como o algoritmo deverá funcionar.

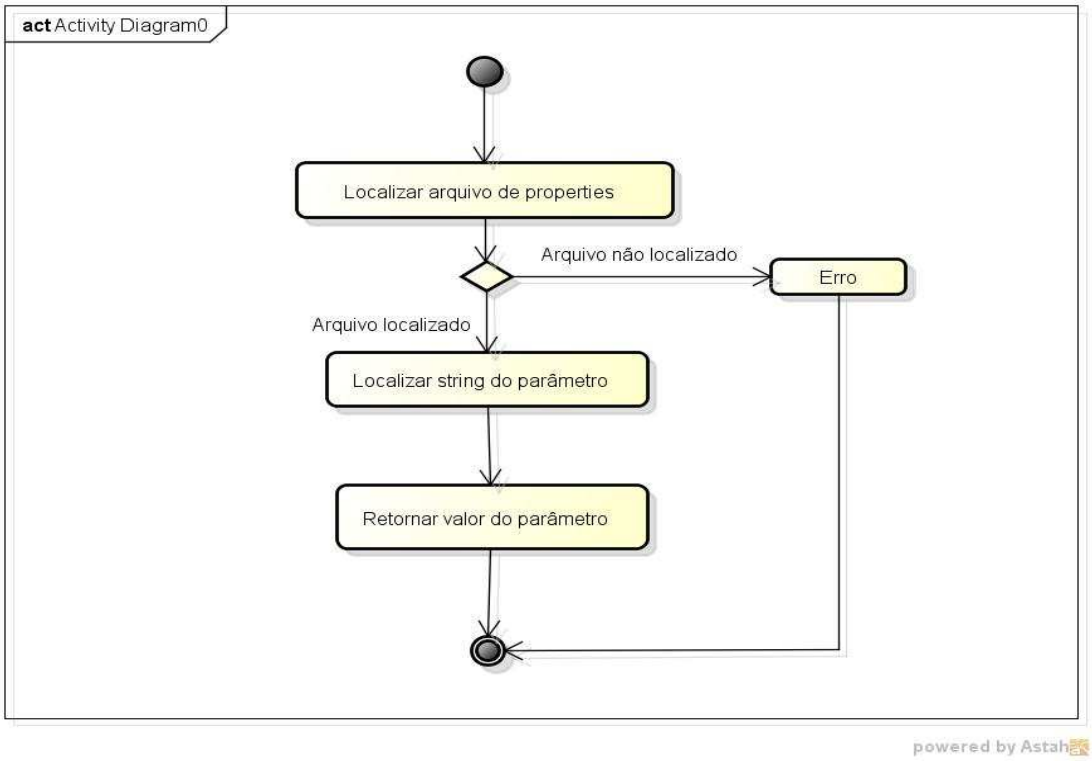

Figura 4. 17 – Diagrama de busca de parâmetros
Fonte: Autor

Portanto, o código do software primeiramente localiza o arquivo na pasta onde o *middleware* está sendo executado. Caso o arquivo não seja localizado, é retornado um erro. Caso seja localizado, o programa busca o valor do parâmetro dentro do arquivo.

O último requisito é o Req17, que solicita um serviço de log. Este requisito foi implementado utilizando um serviço de log baseado na API log4j da apache. Este

serviço utiliza as configurações de um arquivo chamado “log4j.properties” que também deve existir na pasta onde o *middleware* será executado e deve estar devidamente preenchido. Portanto, as duas novas premissas do projeto são: Preq10 – Deve existir o arquivo log4j.properties dentro da pasta onde o middleware for executado, Preq11 – O caminho informado no parâmetro “log4j.appender.fileout.file” deve ser um caminho válido e com acesso total pelo *middleware*. O algoritmo é simples, como visto na figura 4.18.

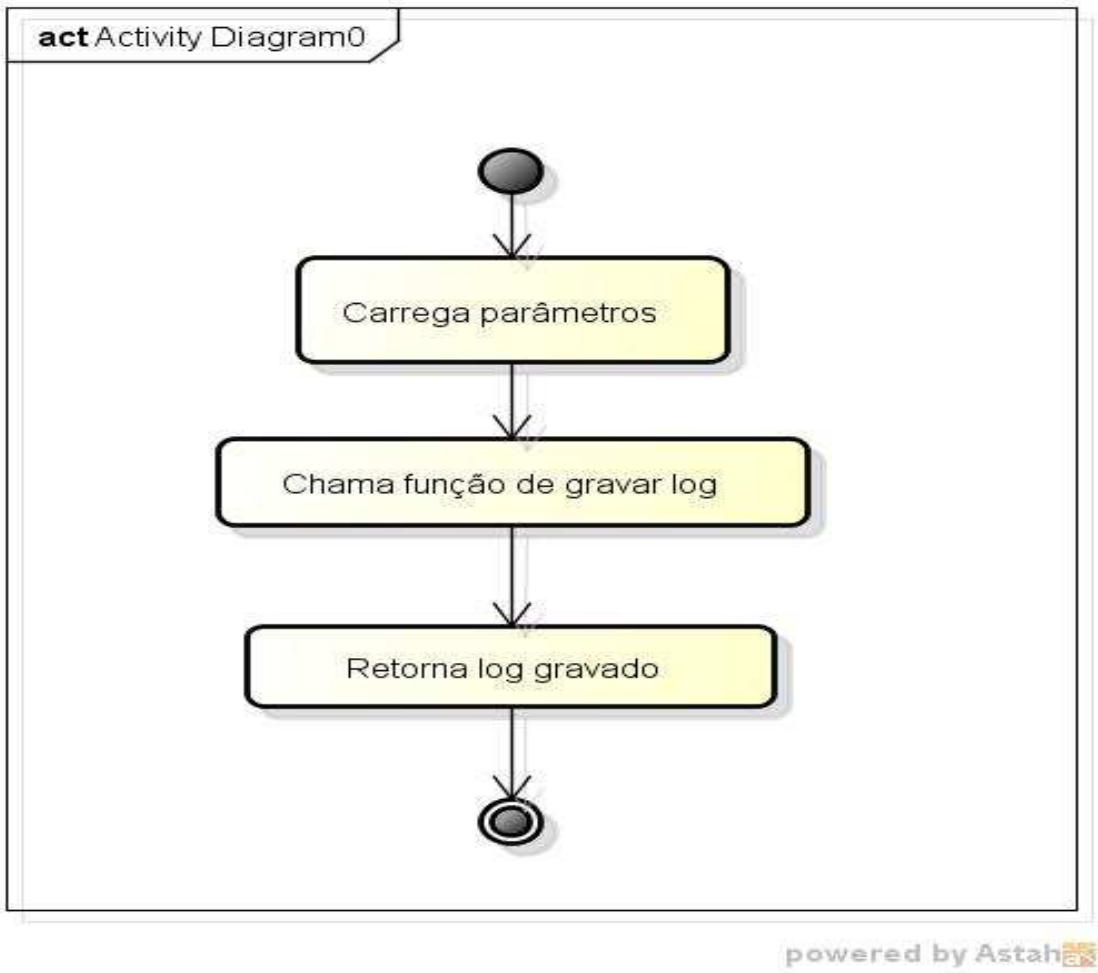

Figura 4. 18 – Diagrama Diagrama Log
Fonte: Autor

O código deve buscar apenas as configurações do arquivo e gravar os logs de acordo com as regras parametrizadas no arquivo *properties*. O log é gravado de acordo com o nível configurado (Info, Debug e Error), para efeito de simplificação, os logs são configurados para modo *debug* em sua maioria, uma vez, que por ser uma aplicação mais simples, não é necessário fazer o controle tão apurado dos logs.

Terminadas as etapas, **Requisito** e **Solução**, foi gerada a tabela da figura 4.19, com os pré-requisitos, a partir da qual foi gerada na seqüência a tabela de rastreabilidade de requisitos visto na figura 4.20.

Agora será iniciada a etapa **Implementação**. Nessa etapa será comentado como o código foi construído, porém os trechos do programa não serão inseridos, uma vez que o código completo é disponibilizado no Apêndice A – Código, deste documento e a visualização do código como um todo é mais simples que trechos cortados.

Preq01	Todo dado enviado dever ser enviado na forma de uma string dividida em 4 partes, a primeira composta do valor da tag ou etiqueta, a segunda do caracter ponto
Preq02	O nome da porta serial, o baudrate da porta serial e o timeout de operação devem ser parametrizados sempre pelo usuário, com valores não nulos
Preq03	O dispositivo USB, deve ser um dispositivo USB HID (Human Interface Device), como teclados, mouses, leitores RFID e Leitores Código de barras comerciais.
Preq04	Existência de um banco de dados Oracle local ou na intranet do usuário
Preq05	Existência de um usuário e senha válidos no banco
Preq06	Todos os dados de tabela e coluna devem ser parametrizados pelo usuário previamente com objetos válidos no banco
Preq07	O usuário do banco deve ter os acessos necessários aos objetos do banco para executar as ações de insert e select
Preq08	Deve existir um arquivo chamado “middleware.properties” dentro da pasta onde o executavel do software for executado
Preq09	Devem ser preenchidos corretamente todos os parâmetros do arquivo de configuração
Preq10	Deve exister o arquivo log4j.properties dentro da pasta onde o middleware for executado
Preq11	O caminho informado no parâmetro “log4j.appender.fileout.file” deve ser um caminho válido e com acesso total pelo middleware

Figura 4. 19 – Tabela de Pré-Requisitos
Fonte: Autor

Req/Preq	1	2	3	4	5	6	7	8	9	10	11
1	◆										
2	◆										
3	◆										
4		◆									
5			◆								
6		◆									
7				◆	◆	◆					
8				◆		◆	◆				
9				◆		◆	◆				
10								◆	◆		
11								◆	◆		
12								◆	◆		
13								◆	◆		
14								◆	◆		
15								◆	◆		
16								◆	◆		
17										◆	◆

Figura 4. 20 – Tabela de Rastreabilidade
Fonte: Autor

Primeiro, é necessário abordar a classe Rotina, onde a package pode ser vista na figura 4.21 e o diagrama da classe na figura 4.22. Nela foram criados 3 métodos: o método **Carga**, o método **Start** e o método **Main**. O método **carga** é responsável por realizar a carga dos parâmetros dentro da memória do middleware, é um método simples que através dos métodos disponibilizados nas classes da package Loader armazena os parâmetros em memória.

O método **start** é responsável por toda implementação do fluxo de dados solicitados nos requisitos Req01, Req02 e Req03. Como demonstrado na figura 4.8, este

algoritmo possui alguns trechos de tomada de decisão e de validação que serão avaliados mais adiante.

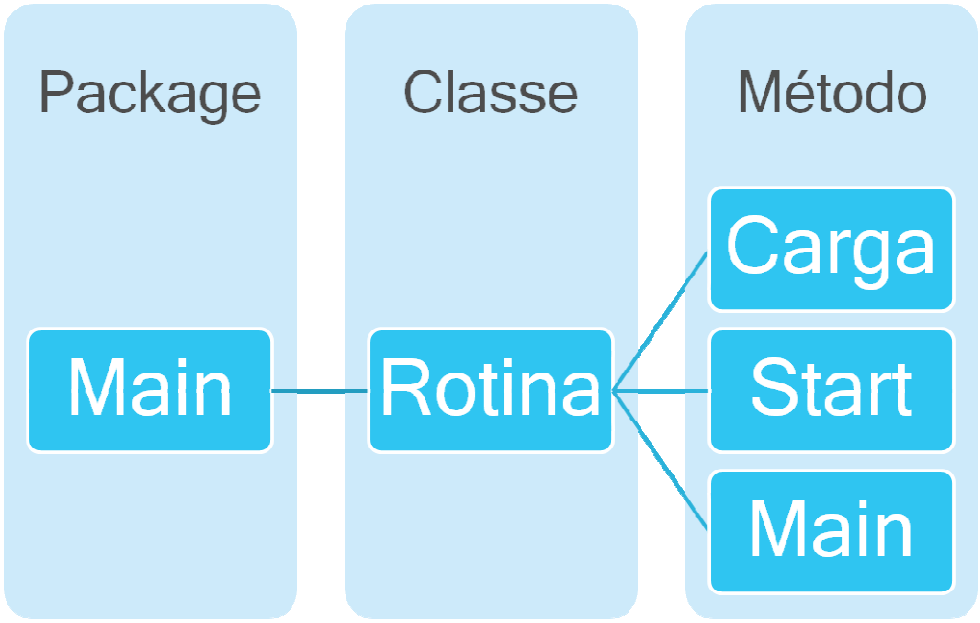

Figura 4. 21 – Package Main
Fonte: Autor

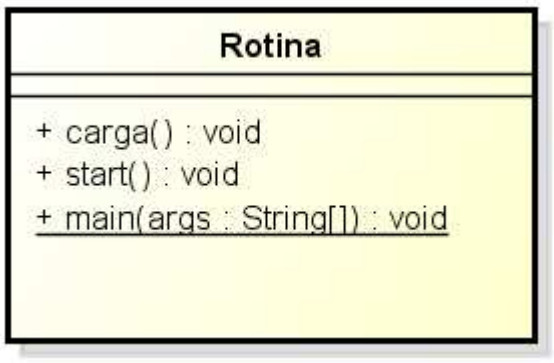

Figura 4. 22 – Classe Rotina
Fonte: Autor

O método **main** é o responsável pela chamada dos 2 métodos anteriores e gerencia o término do programa. Ele foi construído de forma chamar o método **Carga** e em seguida o método **Start,** ao final da execução do método start será questionado ao usuário se o mesmo deseja continua ou não. Caso se deseje continuar, o programa será reiniciado e caso contrário o programa é finalizado.

Voltando para rotina **Start**, esta é responsável por todo fluxo de dados solicitado nos requisitos Req01, Req02 e Req03. Se for observado no diagrama de figura 4.8, este fluxo possui algumas validações e tomadas de decisão importantes.

Primeiro é definido o meio de entrada de dados, através uma cadeia de IF, IF ELSE e ELSE, que utiliza a variável *mode* para receber o valor configurado pelo usuário e compara dentro de cada tomada de decisão. A partir daí, o programa define se o meio de entrada é USB, SERIAL ou nenhum dos dois.

Uma vez identificado o meio, o middleware chama o respectivo método de leitura configurado, *usbreader* ou *serialreader* da classe serialconnector na package serial, e aguarda primeiramente uma entrada de dados. Caso nenhum dado seja retornado para variável *entradadados* o middleware aguarda em loop até que um dado seja capturado pelo método de leitura e retornado na variável *entradadados*.

Assim que o dado for retornado, o software valida se é um dado válido. Para isso ele utiliza o método nativo do Java StringTokenizer para contar quantos tokens existem na string recebida pela variável *entradadados*, caso o número seja 2, o programa irá continuar, caso seja diferente de 2, é retornado que o dado recebido é inválido e o programa retorna para a fase de aguardar entrada de dados.

Após a entrada de um dado válido o programa irá armazenar nas variáveis *Dados1tabela* e *Dados2tabela* da classe VariáveisAmbiente da package loader. O dado da variável *Dados2tabela* é utilizado para identificar se a operação enviada pelo microcontrolador foi INS, REG ou CON e o dado da variável *Dados1tabela* é a tag e/ou etiqueta que é utilizada nas operações do banco.

Dentro do fluxo de cada dado que foi construído dentro de um CASE, o middleware chama o método de operação no banco da classe BancoOperacoes na package BancodeDados, de cada fluxo (AtualizaInsertBanco, AtualizaRegistroBanco e SelectBanco) . Sendo que dentro do fluxo de registro, o software solicita ao usuário a entrada de um dado adicional, que é recebido pela variável *registro*. No fluxo de consulta, após a consulta, o dado é retornado para variável *resposta*.

Caso o usuário envie uma operação não prevista, o programa grava o log e informa ao usuário que a operação enviada não foi prevista, da mesma forma que também caso na opção de consulta ocorra erro ou o dado consultado (tag/etiqueta) não exista na tabela configurada. Em caso de erros no processo de insert das operações Insert e Registro, o software também grava o log e informa ao usuário.

Conforme citado anteriormente, tem-se o suporte de métodos de outras classes, de outros métodos para executar as funcionalidades da classe **Rotina**. Primeiro são analisadas as classes da package Serial. Está package contém 3 classes que juntas implementam os requisitos Req04, Req05 e Req06 e a package pode ser vista na figura 4.23, bem com as classes nas figuras 4.24, 4.25 e 4.26.

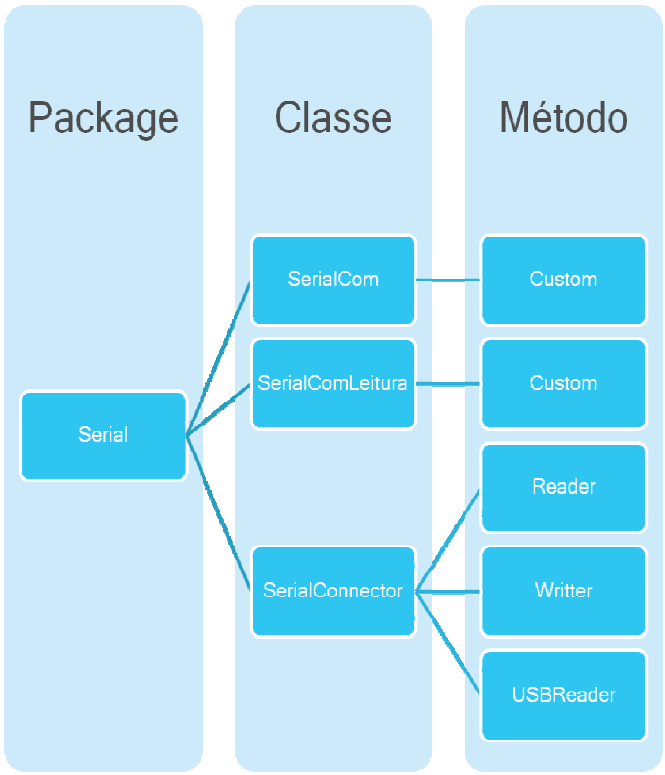

Figura 4. 23 – Package Serial
Fonte: Autor

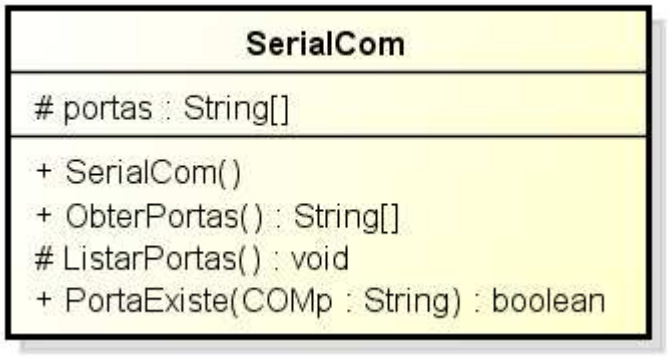

Figura 4. 24 – Classe SerialCom
Fonte: Autor

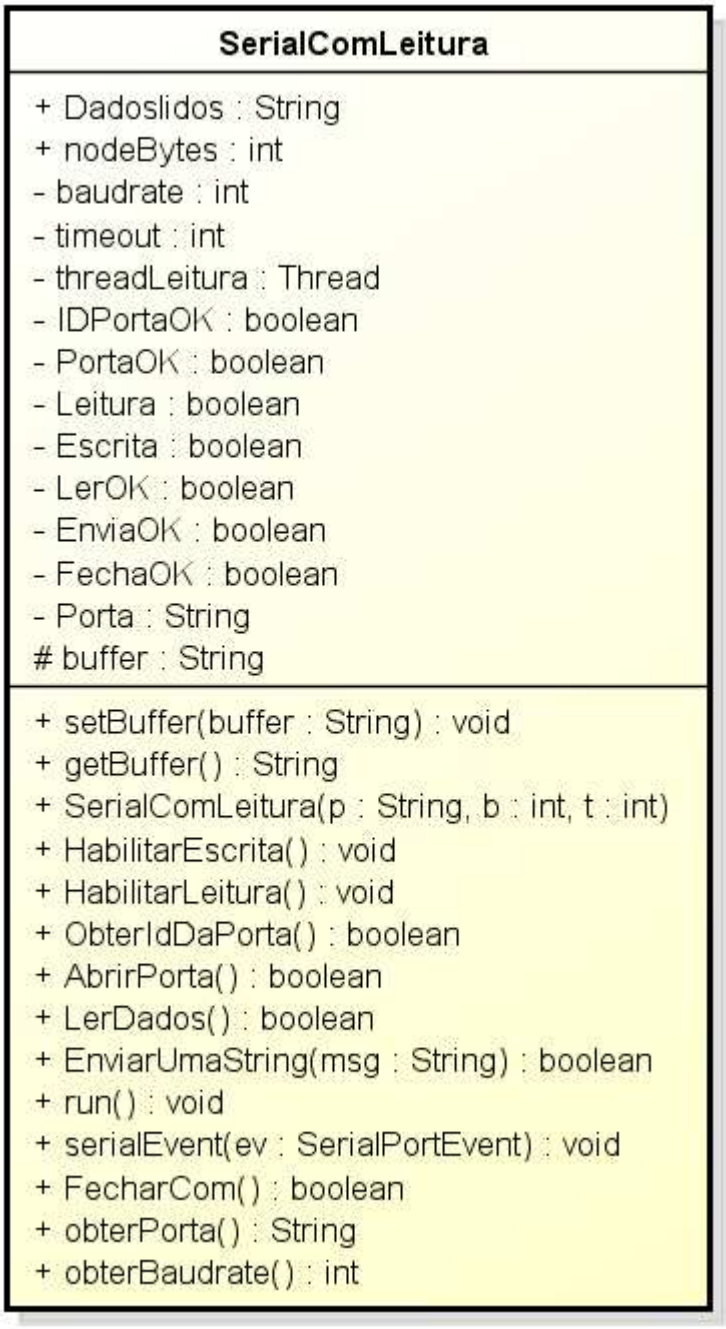

Figura 4. 25 – Classe SerialComLeitura
Fonte: Autor

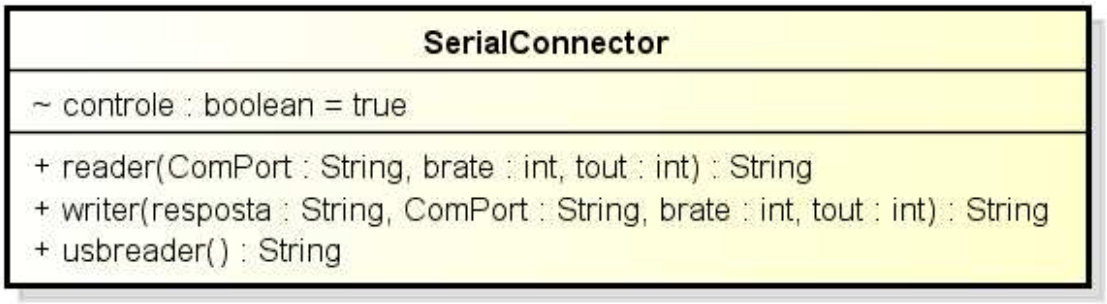

Figura 4. 26 – Classe SerialConnector
Fonte: Autor

A principal classe da package é a *SerialConnector*, dentro dela estão os métodos responsáveis para leitura na USB, leitura na Serial e escrita na Serial. Esta classe utiliza os métodos da classe *SerialComLeitura*, que por sua vez estende os métodos da classe *SerialCom*.

As classes *SerialCom* e *SerialComLeitura* foram construídas e modificadas a partir do artigo "Utilizando a API RXTX para manipulação da serial – www.devmedia.com.br". Estas 2 classes abordam as funcionalidades básicas de comunicação com porta serial, como localizar uma porta no sistema operacional, abrir a comunicação com a porta , enviar, receber dados e etc. Elas foram construídas utilizando a API RXTX que uma API de comunicação serial aberta.

A classe *SerialConnector*, como informado, é a classe principal da package. Esta classe agrupa e ordena os métodos da classe *SerialComleitura* e é onde foi implementado os diagramas das figuras 4.13, 4.14 e onde também foi criado o método do diagrama da figura 4.15.

Os métodos *reader* e *writter* são similares, ambos utilizam um método de segurança da classe *SerialComLeitura* para evitar que o programa envie e receba dados ao mesmo tempo (*HabilitarLeitura* ou *HabilitarEscrita*), na sequência, ambos chamam o método *ObterIdDaPorta* para localizar se a porta configura existe no hardware onde o middleware é executado. O próximo método chamado é o *AbrirPorta*, que realiza a abertura da porta serial para comunicação. Após estas etapas iniciais, que são similares, cada método chama o método especifico da operação realizada (*LerDados* ou *EnviarUmaString*) e por último, ambos chamam o método *FecharCom*.

O método *usbreader*, assim como descrito no diagrama da figura 4.15, apenas recebe um dado via entrada de dados convencional. Portanto, o software apenas atribui

os valores inseridos pelo usuário no vetor *bytes* e depois converte em uma string retornada na variável *buffer*.

Dentro da package *BancodeDados*, temos a classe *BancoOperacoes*. Nesta classe estão implementados os recursos que atendem ao diagrama da figura 4.16 e aos requisitos Req07, Req08 e Req09, podendo a package ser observada na figura 4.27 e a classe na figura 4.28.

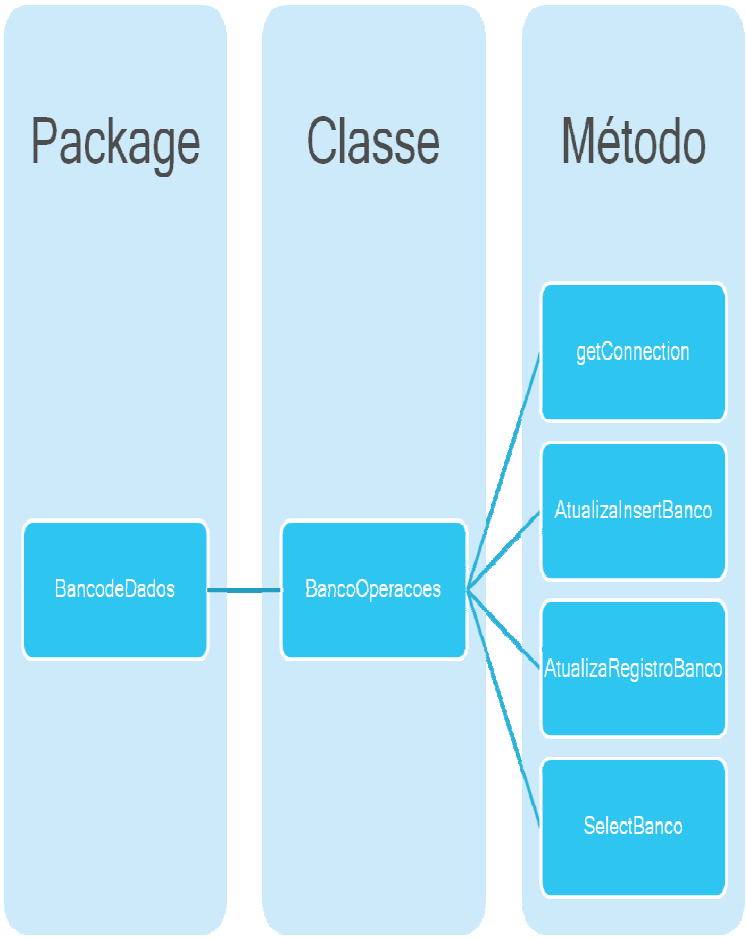

Figura 4. 27 – Package BancodeDados
Fonte: Autor

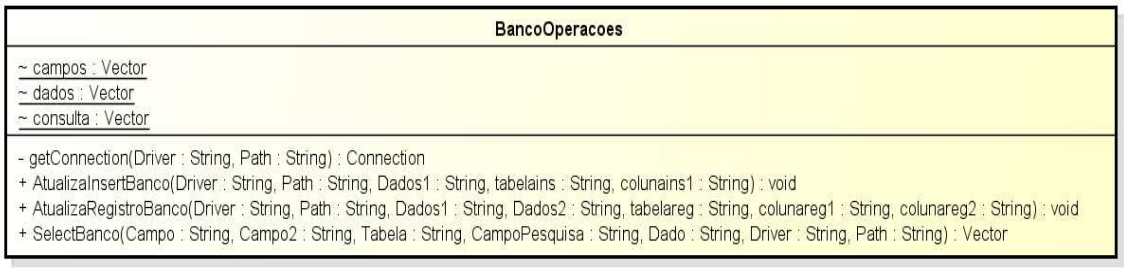

Figura 4. 28 – Classe BancoOperacoes
Fonte: Autor

Está classe possui 4 métodos. O primeiro método é o *getConnection*, que é responsável pela conexão com o banco de dados. O segundo método é o *AtulizaInsertbanco* que realiza o insert no banco para o fluxo de dados Insert. Nosso terceiro método, *AtualizaRegistroBanco*, é responsável pelo insert no banco quando o fluxo é Registro. Por último, temos o método SelectBanco, que realiza o select no banco de dados.

Os métodos *AtulizaInsertbanco*, *AtualizaRegistroBanco* e *SelectBanco* utilizam o método *getConnection* para realizar a conexão com o banco e armazenar na variável *conBD*, após isso cada uma das operações monta sua instrução sql própria e executa no banco. A montagem da instrução sql utiliza os parâmetros configurados pelo usuário de tabelas e colunas.

Os requisitos do número 10 ao 17 solicitam que seja possível ao usuário parametrizar determinadas variáveis de ambiente, variáveis que foram utilizadas nas outras classes já apresentadas. Como foi definido, a parametrização ocorrerá em um arquivo *properties* que é carregado através da classe *GetProperties*, exibida na figura 4.29, que é integrante da *package Loader*, exibida na figura 4.30.

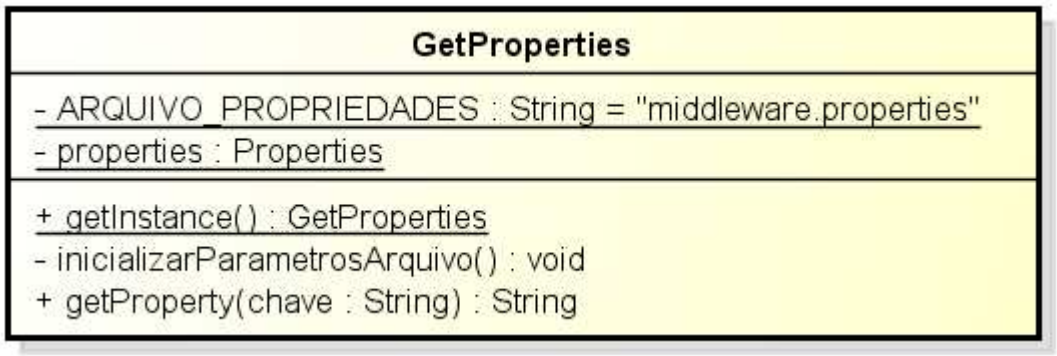

Figura 4. 29 – Classe GetProperties
Fonte: Autor

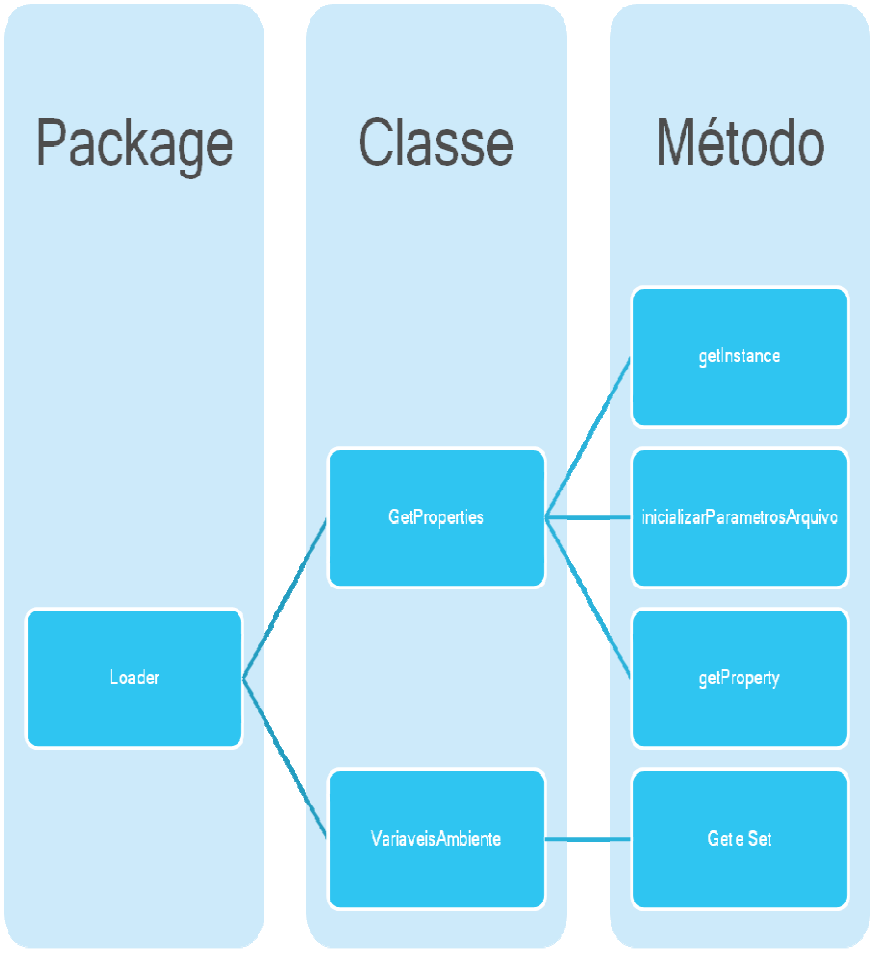

Figura 4. 30 – Package Loader
Fonte: Autor

Está classe implementa o fluxo do diagrama na figura 4.17. Para facilitar a manipulação dos valores que esta classe implementa, foi criado a classe *VariaveisAmbiente* que apenas executa as funções de Set e Get das variáveis carregadas do arquivo e armazenadas na memória, motivo pelo qual o autor não considerou necessário gerar o diagrama da classe. O método carga da classe rotina, implementada a chamada do método *getProperty* da classe *GetProperties* passando o valor recebido para o método set de cada respectiva variável na classe VariaveisAmbiente.

O caminho do arquivo properties fica armazenado no código do método e está configurado para o diretório onde o programa for executado e com o nome de "*middleware.properties*".

O Rep17 demanda a criação de logs das atividades executadas e como descrito no diagrama da figura 4.18, este algoritmo é simples, uma vez que foi utilizado a api Log4J da Apache e a package pode ser vista na figura 4.31, bem como a classe na figura 4.32.

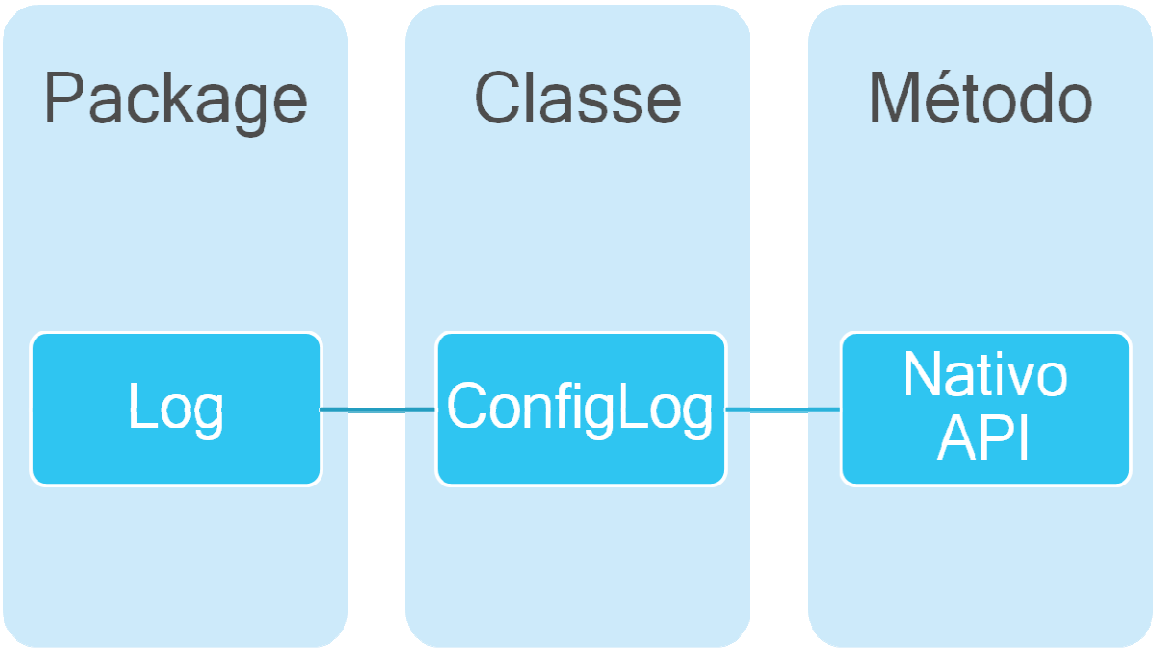

Figura 4. 31 – Package Log
Fonte: Autor

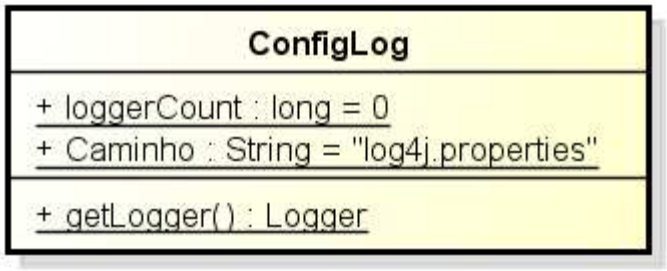

Figura 4. 32 – Classe ConfigLog
Fonte: Autor

Para gravar qualquer log, basta inserir a chamada do método *getLogger* da classe ConfigLog passando como parâmetro o nível do log desejado (INFO, DEBUG ou ERROR). Isto é, o quando um log for configurado como debug, ele somente será registrado no log se o nível do log configurado na chamada do método *SetLevel* da própria API log4j for daquele nível ou acima.

O caminho onde o arquivo log4j.properties é buscado fica configurado no código e está configurado para a pasta onde o usuário executar o software. Dentro do arquivo log4j.properties é possível configurar o nome do arquivo de log gerado e a pasta onde ele será salvo.

Com o final da etapa **Implementação** tem-se a etapa **Validação**, porém esta etapa será tratada apenas no capítulo 5.

CAPÍTULO 5 – APLICAÇÃO DA SOLUÇÃO COM RESULTADOS

5.1 – Apresentação da área de aplicação do modelo

O middleware desenvolvido é voltado para integração de arquiteturas que recebam dados, no padrão proposto na figura5.1, na porta serial ou na porta USB via uma interface HID (*Human Interface Device*), dentro de um ambiente operacional Windows de 32 bits e tenham uma integração via conexão local ou dentro da intranet com um banco de dados Oracle instalado em qualquer plataforma.

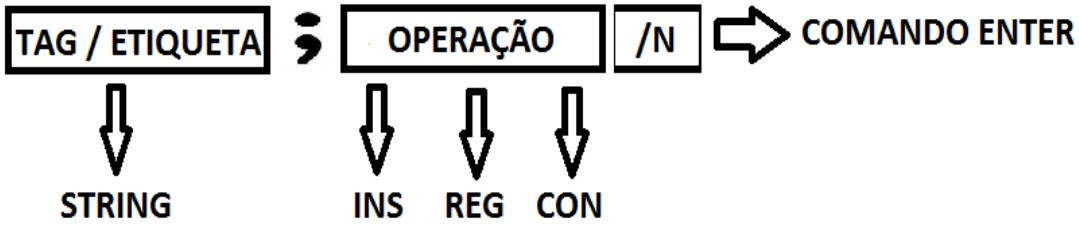

Figura 5. 1– Padrão Proposto
Fonte: Autor

O software pode ser aplicado, com todas ou em parte de suas funcionalidades, em projetos, tanto de sala de aula, quanto em projetos de graduação, que sejam baseados em aquisição, tratamento e integração de dados de etiquetas, tags ou labels.

Qualquer arquitetura, exposta em sala de aula ou planejada como projeto de conclusão de curso, que utilize ou simule tecnologias de RFID e/ou Código de barras, pode ser integrada com banco de dados utilizando o aplicativo.

Ele permite, por exemplo, a construção, durante o curso de microcontroladores ou em propostas de projetos acadêmicos, de protótipos de projetos comerciais baseados em tags, etiquetas ou labels, como projetos implementados em lojas de controle de estoque através do código de barras, controle de identificação e acesso em prédios baseados em RFID, controle de tráfego estruturado em telemetria RFID e outros.

A ferramenta pode ser aplicada em qualquer projetos que utilize os três tipos de regra de negócio primitivos propostos (Insert, Registro e Consulta), ficando a cargo do usuário a criatividade de como utilizar a ferramenta.

5.2 – Descrição da aplicação do modelo

Para aplicação do software é necessário validar alguns requisitos técnicos, que podem ser de hardware ou software.

O requisito de hardware necessário é um desktop ou notebook com uma porta serial funcional, seja ela virtual ou real.

Os requisitos de software é que o ambiente do desktop/notebook utilizado seja uma versão do Windows de 32 bits com a Java Virtual Machine instalada e um banco Oracle instalado localmente ou na intranet, em qualquer plataforma de sistema operacional.

Deve-se inicialmente copiar a DLL "rxtxSerial.dl" para o diretório “C:\Windows\System32”, assim como na figura 5.2.

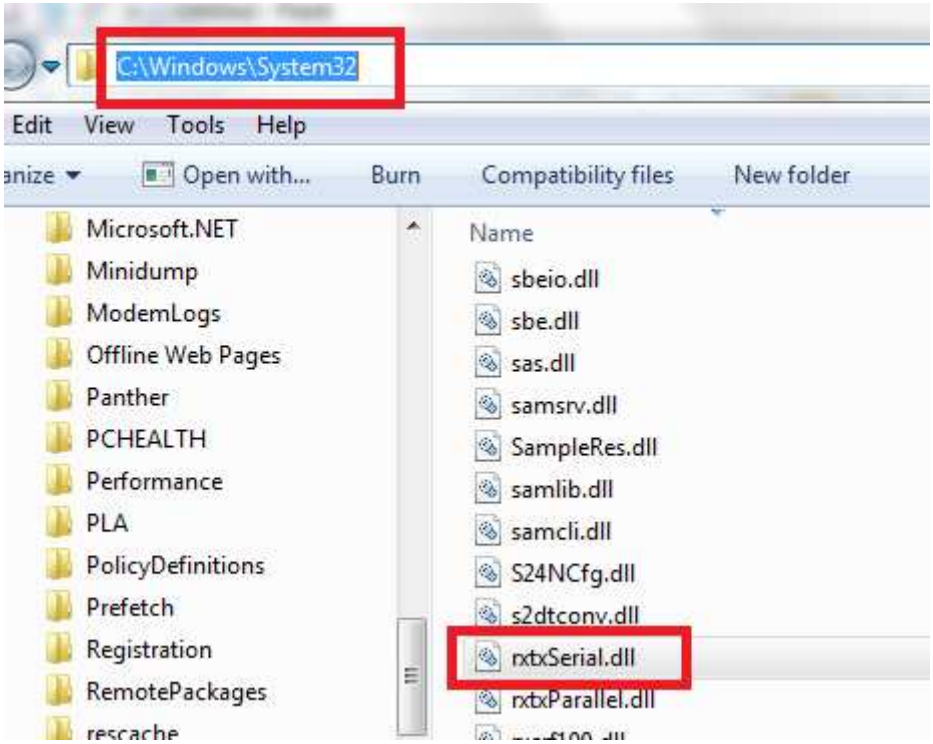

Figura 5. 2– DLL
Fonte: Autor

Em seguida deve se copiar o arquivo "RXTXcomm.jar" para dentro da pasta “Lib” do diretório onde a Java Virtual Machine foi instalada, como no exemplo de figura 5.3.

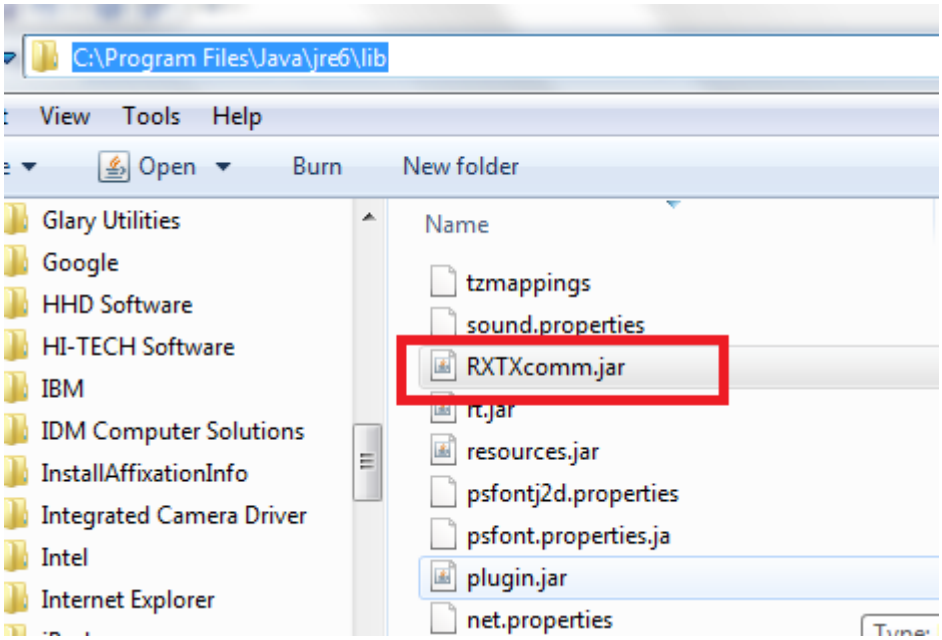

Figura 5. 3– Lib
Fonte: Autor

Após a extração do arquivo “middleware.zip” em qualquer pasta do desktop, deve ser aberto o arquivo “middleware.properties”. Nesse arquivo deve ser configurado os parâmetros customizáveis da ferramenta, como driver do banco de dados, path de conexão com o banco de dados, tabelas de insert, registro e consulta, colunas de insert, registro e consulta, tempo de sleep da porta serial em modo leitura, tempo de sleep da porta serial na abertura, porta serial utilizada, baudrate da porta, mode de entrada dos dados e etc.

Após as configurações citadas acima, deve ser executado, via CMD, na pasta onde foi descompactado o arquivo “middleware.zip” o comando “java –jar Middleware.jar”, conforme a figura 5.4.

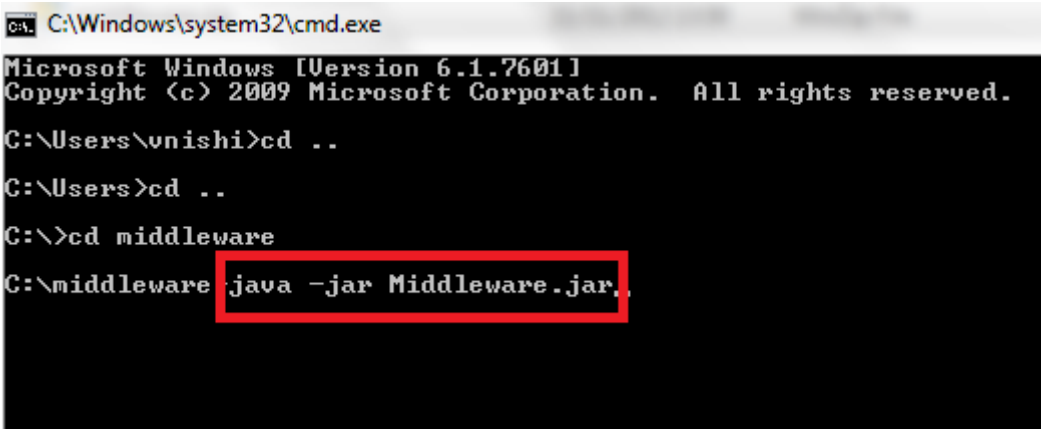

Figura 5. 4– Exe
Fonte: Autor

Uma vez iniciado o executável, é exibida uma mensagem informando que a carga das variáveis configuradas no arquivo "*middleware.properties*" foi realizada, como pode ser observado na figura 5.5.

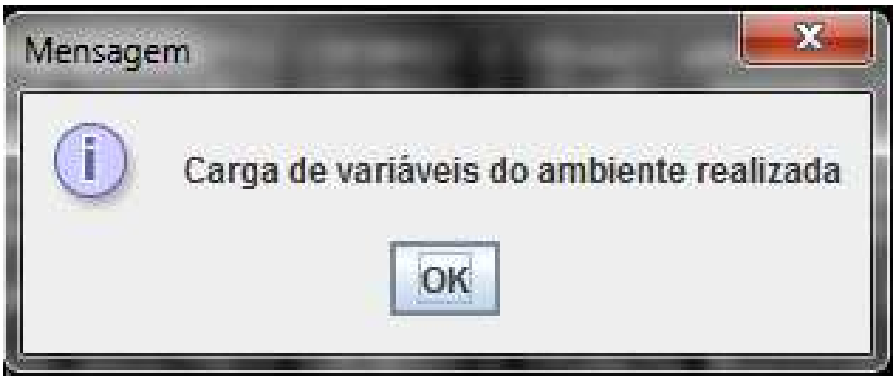

Figura 5. 5–Carga
Fonte: Autor

Após a carga será definido o modo de entrada de dados configurado no arquivo "middleware.properties" no parâmetro modoentrada, que pode ser Serial ou USB e informado o modo de entrada escolhido, conforme exemplo da figura 5.6.

Figura 5. 6– Modo
Fonte: Autor

Uma vez escolhido o modo, é exibida a tela da figura 5.7, periodicamente, informando que o aplicativo está aguardando a entrada de dados. O ciclo se repete de acordo com o tempo em milissegundos configurado no parâmetro “sleep” do arquivo “middleware.properties”.

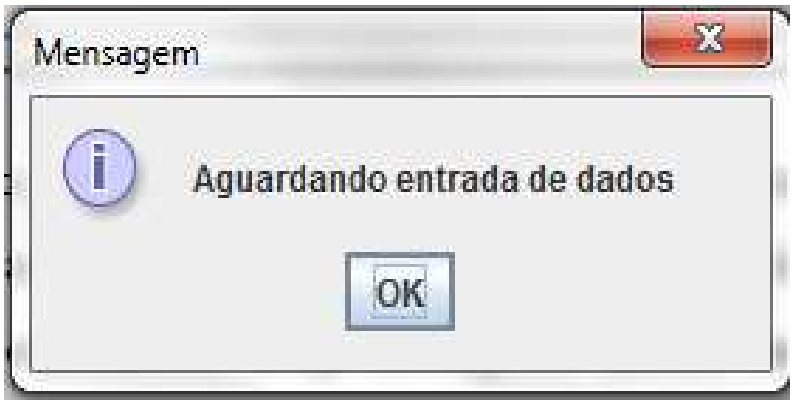

Figura 5. 7– Entrada Dados
Fonte: Autor

O loop da entrada de dados somente será interrompido quando algum dado for recebido no meio de entrada de dados selecionado. Este meio de entrada deve ser via

porta Serial ou USB. Quando for Serial o software fará um poolling de leitura na porta serial, aguardando um dado seja enviado de um microcontrolador ou via software de simulação com /n ou /r no final. Na opção de USB, é aguardado uma string via interface USB HID, ou seja é uma entrada de dados normal, similar a uma entrada de teclado convencional, portanto deve ser dado ENTER ao final do Input. Uma vez recebido um dado será exibida a mensagem da figura 5.8.

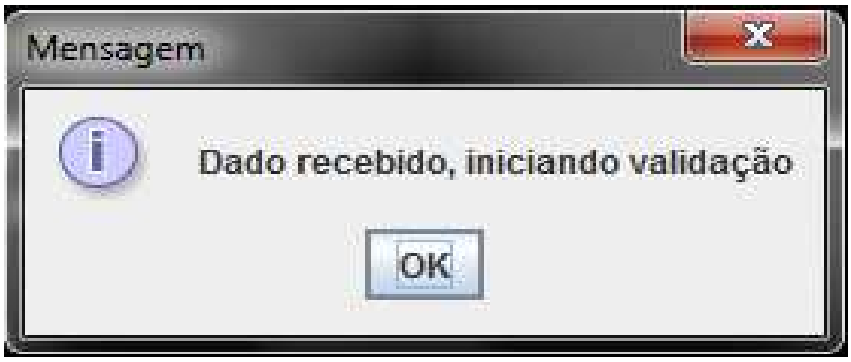

Figura 5. 8– Recebimento
Fonte: Autor

O dado é validado, de acordo com a estrutura informada na Figura01, e caso tenha sido enviado no padrão, o programa grava no arquivo de log que o dado não é válido e retorna para o loop de aguardando dados. Caso o dado seja válido, é definido da operação, que pode ser **INSERT**, **REGISTRO** ou **CONSULTA**, conforme figura 5.9.

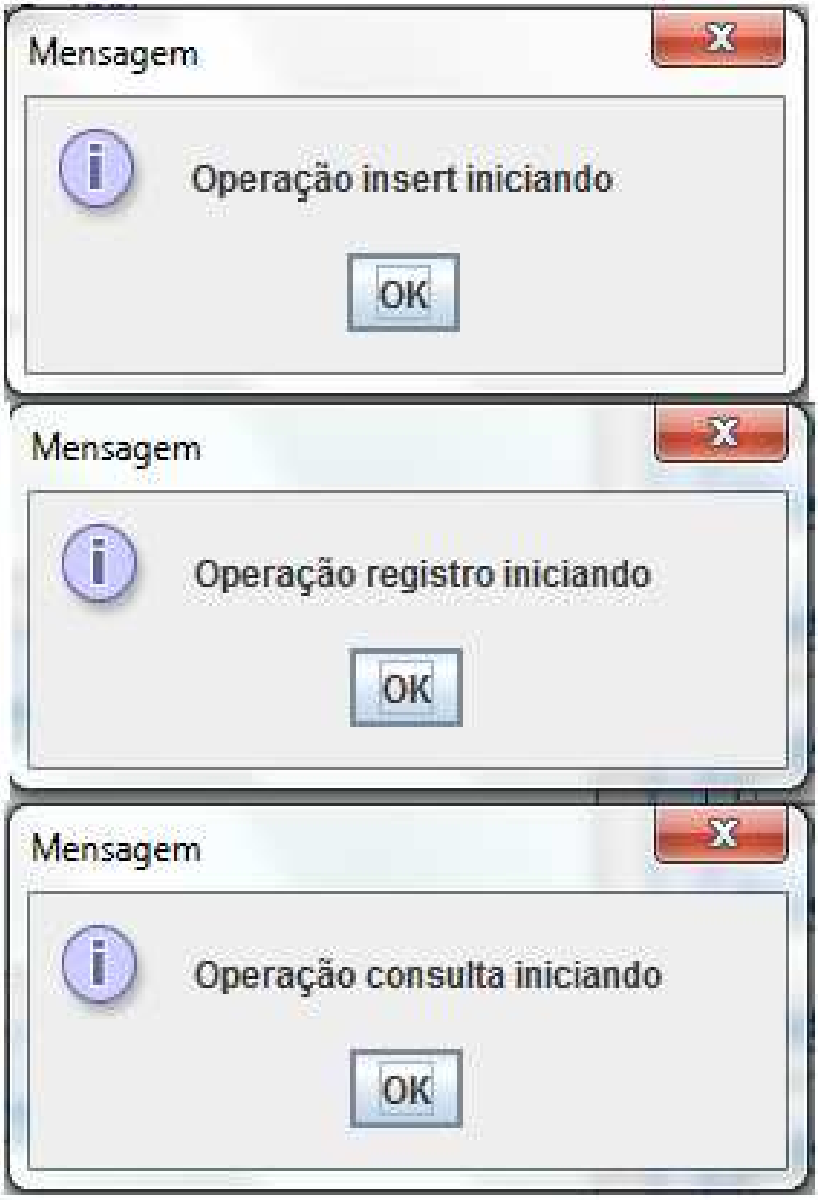

Figura 5. 9– Operação
Fonte: Autor

Para operação de Insert, o dado é inserido na tabela e coluna informados nos parâmetros tabelainsert e colunainsert1 do arquivo “middleware.properties”. Caso a operação tenha sucesso é exibida a tela da figura 5.10 e em caso de erro, o erro é gravado no arquivo de logs e em seguida apresentado a tela da figura 5.10.

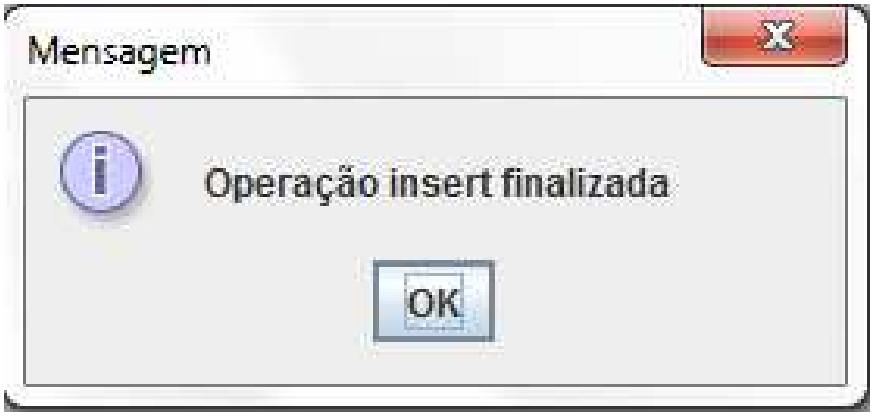

Figura 5. 10– Insert
Fonte: Autor

Na operação Registro, após definida a operação, é exibida uma tela solicitando a entrada de um dado adicional pelo próprio usuário, como visto na figura 5.11.

Figura 5. 11– Registro Adicional
Fonte: Autor

A tela da figura 5.11, é exibida até que o usuário insira um dados não nulo. Após receber o dado, o dado recebido na porta serial ou USB e o dado fornecido na tela da figura 5.11 são inseridos na tabela e colunas parametrizados nos parâmetros tabelaregistro, colunaregistro1 e colunaregistro2.

Ao final é exibida a mensagem apresentada da figura 5.12 em caso de sucesso e no caso de erro, também é exibido a mensagem da figura 5.12, mas o erro é gravado no arquivo de log.

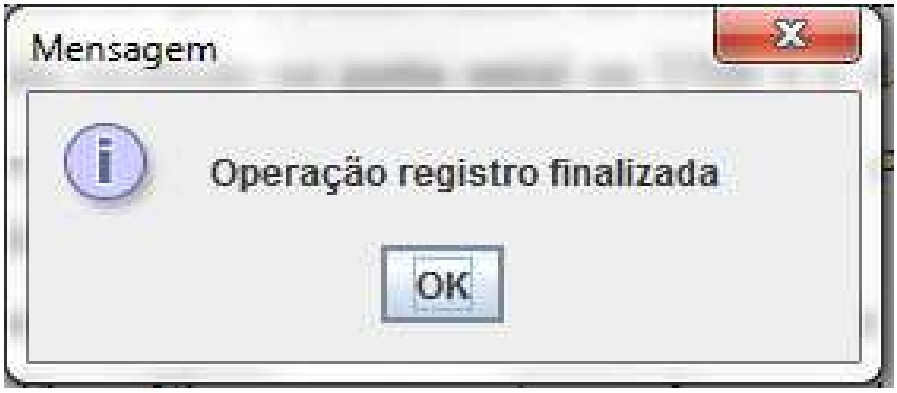

Figura 5. 12– Registro
Fonte: Autor

Quando a operação for Consulta, o dado enviado é consultado na tabela informada no parâmetro tabelaconsulta e o dado é consultado na coluna do parâmetro campopesquisa do arquivo “middleware.properties” e os dados retornados devem ser configurados no parâmetros campo1tabela e campo2tabela.

O dado retornado pela consulta do parâmetro campo1tabela, é o dado enviado na porta serial. Ao final é exibido a tela apresentada na figura 5.13 e em caso de erro ou não localizar o dado na tabela, também é registrado o log do ocorrido.

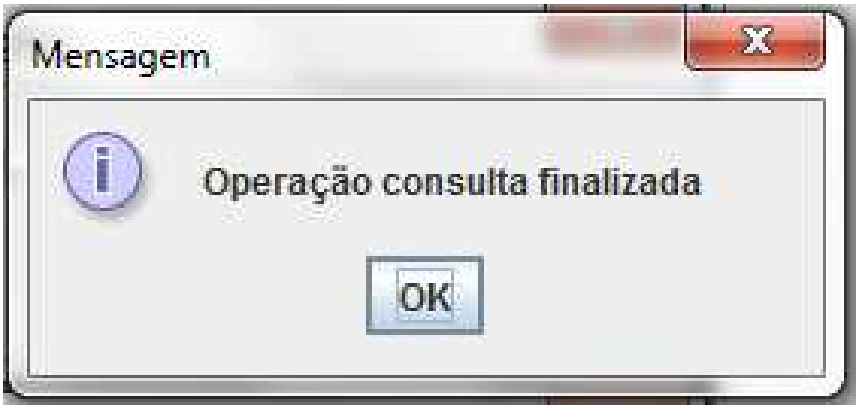

Figura 5. 13– Consulta
Fonte: Autor

Sempre ao final de qualquer uma das três operações, é exibido a tela da figura 5.14, questionando se deseja encerrar o aplicativo ou reiniciar o mesmo.

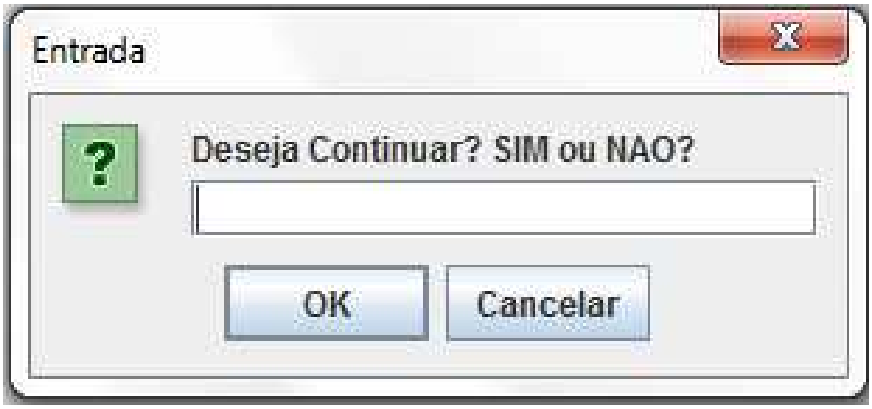

Figura 5. 14– Deseja Continuar
Fonte: Autor

Este é o modo de funcionamento do software, com base nele pode-se preparar a arquitetura do projeto acadêmico para utilizar as funcionalidades citadas acima, de acordo com a necessidade do projeto.

Para isso deve o projeto deve primeiro definir quais as operações serão necessárias, projetos de monitoramento de tráfego, como posicionamento de veículos, trajetos e etc, por exemplo, precisam apenas da operação de Insert, para registrar onde a Tag do veículo foi monitorada. No caso de projetos que simulem um mercado ou loja, seria necessário as operações de registro, para registrar o código de barras para o produto e a operação de consulta, para verificar o preço da mercadoria vinculado ao código, também por exemplo.

Uma vez definido as operações necessárias, o usuário deve modelar seu banco de dados e configurar o arquivo de propriedades da ferramenta, na seqüência, deve ser conectado o microcontrolador com leitor RFID, Código de Barras ou apenas simular a leitura, enviado os dados para o desktop na porta serial ou USB e esperando um retorno, caso necessário.

5.3 – Avaliação global do modelo proposto

Neste ponto do trabalho será realizada a última etapa do projeto proposto no capítulo 4, que é a etapa Validação.

São testados se todos os requisitos propostos anteriormente foram atendidos e evidenciar os testes, dentro das premissas estabelecidas.

No tópico anterior, já foi exibido algumas telas de aplicação o que seria o suficiente para considerar que parte dos requisitos já estariam implementados, porém ficaria um

tanto quando confuso identificar quais seriam esses requisitos, com isso, iremos criar Casos de Testes e evidenciar, tanto com telas, quanto com logs os resultado esperados.

Para evidenciar a aplicação são utilizados 2 softwares de apoio, o VSPE e o RcomSerial. O VSPE é um software que permite criar portas paralelas virtuais no notebook, esse software é necessário, pois o notebook, onde o projeto foi desenvolvido, não possui portas seriais e o equipamento a ser utilizado no curso de microcontroladores ainda não estava disponível quando os testes do middleware foram realizados. O software RcomSerial é um software que envia e recebe dados na porta serial, esse software é necessário, pois simula a função de um microcontrolador ao enviar e receber dados.

A seguir são apresentados os casos de testes, cada caso é vinculado a um ou mais requisito dentro do caso serão descritos os passos. Os casos de teste são:

1 - Enviar tag na porta serial com a operação INS (Req01, Req04, Req07, Req09, Req10, Req11, Req12, Req13, Req14, Req15, Req16 e Req17)

1.1 – Parametrizar arquivo middleware.properties;

1.2 – Validar que dado não existe no banco

1.3 – Iniciar Middleware;

1.4 – Enviar dado;

1.5 – Pesquisar dados no banco.

Evidência: Apêndice B – Evidências Caso 1

2 - Enviar tag na porta USB com a operação INS (Req01, Req05, Req07, Req09, Req10, Req11, Req12, Req13 e Req17)

1.1 – Parametrizar arquivo middleware.properties;

1.2 – Validar que dado não existe no banco

1.3 – Iniciar Middleware;

1.4 – Enviar dado;

1.5 – Pesquisar dados no banco.

Evidência: Apêndice C – Evidências Caso 2

3 - Enviar tag na porta serial com a operação REG (Req02, Req04, Req07, Req09, Req10, Req11, Req12, Req13, Req14, Req15, Req16 e Req17)

1.1 – Parametrizar arquivo middleware.properties;

1.2 – Validar que dado não existe no banco

1.3 – Iniciar Middleware;

1.4 – Enviar dado;

1.5 – Acrescentar dado adicional;

1.6 – Pesquisar dados no banco.

Evidência: Apêndice D – Evidências Caso 3

4 - Enviar tag na porta USB com a operação REG (Req02, Req05, Req07, Req09, Req10, Req11, Req12, Req13 e Req17)

1.1 – Parametrizar arquivo middleware.properties;

1.2 – Validar que dado não existe no banco

1.3 – Iniciar Middleware;

1.4 – Enviar dado;

1.5 - Acrescentar dado adicional;

1.6 – Pesquisar dados no banco.

Evidência: Apêndice E – Evidências Caso 4

5 - Enviar tag na porta serial com a operação CON (Req03, Req04, Req6, Req07, Req08, Req10, Req11, Req12, Req13, Req14, Req15, Req16 e Req17)

1.1 – Parametrizar arquivo middleware.properties;

1.2 – Validar que dado existe no banco

1.3 – Iniciar Middleware;

1.4 – Enviar dado;

1.5 – Validar que dado retornou na serial.

Evidência: Apêndice F – Evidências Caso 5

6 - Enviar tag na porta USB com a operação CON (Req03, Req05, Req06, Req07, Req08, Req10, Req11, Req12, Req13, Req14, Req15, Req16 e Req17)

1.1 – Parametrizar arquivo middleware.properties;

1.2 – Validar que dado existe no banco

1.3 – Iniciar Middleware;

1.4 – Enviar dado;

1.5 – Validar que dado retornou na serial.

Evidência: Apêndice G – Evidências Caso 6

Os Casos acima contemplam todos os requisitos propostos e evidenciam que os mesmo forma implementados com êxito, dentro das premissas propostas.

CAPÍTULO 6 – CONCLUSÃO

6.1– Conclusões

Após a avaliação do problema, da solução sugerida, dos requisitos levantados, da implementação do código e da validação do software, pode-se concluir que, dentro dos limites do escopo, o aplicativo desenvolvido atendeu as expectativas.

A solução foi implementada com sucesso e os requisitos propostos foram implementados e validados um a um com êxito, desde que sigam as premissas do propostas no desenvolvimento.

Os objetivos do trabalho, propostos inicialmente foram implementados e explicados no capítulo 4 e a validação dos mesmos ocorreu no capítulo 5.

O problema apresentado é extremamente amplo e possui diversas necessidades e lacunas a serem preenchidas, a solução apresentada não irá sanar todas estas lacunas. Porém, ela irá atender parte das necessidades, abrindo as portas e servindo de base para outros projetos de mesmo intuito.

O projeto possui como pontos fortes a portabilidade e adaptabilidade do software para diversas arquiteturas de soluções RFID e código de barras existentes no mercado, permitindo a construções de exemplos mais próximos da realidade, tanto em quanto de projetos acadêmicos. Em contrapartida, a solução apresentada limita o usuário dentro dos 3 fluxos de dados propostos e ao padrão de comunicação, obrigando o usuário a se adaptar ao *middleware*.

Pode-se concluir que apesar do proposto no projeto ter sido concluído com sucesso, dentro das premissas estabelecidas, o problema apresentado foi resolvido apenas parcialmente, uma vez que ainda existem outras áreas e outros projetos que necessitam de uma integração, mas que não foram objetivos deste projeto.

Na figura 6.1, pode ser observado as conclusões agrupadas.

Figura 6. 1– Conclusões
Fonte: Autor

6.2– Sugestões para trabalhos futuros

O *middleware* desenvolvido é aplicável à integração de microcontroladores, utilizando tecnologias de RFID ou leitura de código de barras e um banco de dados, isso preenche parte das necessidades de projetos acadêmicos, porém existem outros pontos carentes de integração.

Como sugestão para trabalhos futuros, tem-se:

1) Expandir a aplicação do *middleware*, criando uma plataforma de integração, que permita ao usuário construir seu próprio serviço, com seu padrão próprio de comunicação e regras de negócio. Ou seja, um software que não prenda o usuário as 3 operações definidas neste projeto e a string de comunicação proposta e sim permita que o usuário crie seu fluxo de dados próprio e seu padrão de comunicação, assim como em middlewares comerciais, como exemplo barramentos SOA ou middleware Vitria.

2)Outra sugestão é expandir a comunicação para escrita também na porta USB, uma vez que isto tornaria o middleware aplicável a soluções comerciais de pequeno e médio porte, já suportaria entrada e saída de dados tanto na serial, quanto USB, o que torna o middleware mais flexível e menos dependente do hardware do usuário.

REFERÊNCIAS

API RXTX, http://rxtx.qbang.org/wiki/index.php/Main_Page, acessado dia 14/12/2012.

George Coulouris, Tim Kindberg, Jean Dollimore. **Sistemas Distribuídos : Conceitos e Projeto**, Bookman, 4ºed, 2007.

H.M. Deitel, P.J. Deiteil. **Java, Como Programar**, Bookman, 3º Ed, 2001.

Petter Rafael Villa Real, **Utilizando a API RXTX para manipulação da serial**, Silva, www.devmedia.com.br, acessado dia 14/12/2012.

Pressman R. **Engenharia de Software**, McGraw-Hill Interamericana, 6ª Ed, 2005.

Project Management Institute. **Um Guia do Conhecimento em Gerenciamento de Projetos (Guia PMBOK)**, PMI, 4º ED, 2009.

APÊNDICE A - Código

Package Main

Classe Rotina

```
/* Rotina principal responsável por carregar as variáveis do ambiente e chamar a main */

package Main;

import java.util.StringTokenizer;
import java.util.Vector;

import javax.swing.JOptionPane;

import org.apache.log4j.Level;
import org.apache.log4j.Logger;

import BancodeDados.BancoOperacoes;
import Loader.GetProperties;
import Loader.VariaveisAmbiente;
import Log.ConfigLog;
import Serial.SerialConnector;

public class Rotina {

	private static GetProperties getp = new GetProperties();
	private static final Logger log = ConfigLog.getLogger();
	static VariaveisAmbiente vamb = null;
	BancoOperacoes bo = new BancoOperacoes();
	SerialConnector serial = new SerialConnector();

	// Método responsável pela carga de variáveis do ambiente.
	public void carga() throws Exception{

		GetProperties instancia = getp.getInstance();
		vamb = new VariaveisAmbiente();

		vamb.setNivel_log(instancia.getProperty("nivel_log"));
		log.setLevel(Level.toLevel(vamb.getNivel_log()));

		log.info("Nível de LOG setado para: " + vamb.getNivel_log());
		log.info("");

		log.info("");
		log.info("Setando variáveis iniciais do módulo...");
		log.info("");

		vamb.setDriver(instancia.getProperty("driver"));
		vamb.setPath(instancia.getProperty("path"));
		vamb.setTabelacon(instancia.getProperty("tabelaconsulta"));
		vamb.setTabelains(instancia.getProperty("tabelainsert"));
		vamb.setTabelareg(instancia.getProperty("tabelaregistro"));
		vamb.setCampo1tabela(instancia.getProperty("campo1tabela"));
		vamb.setCampo2tabela(instancia.getProperty("campo2tabela"));
		vamb.setCampopesquisa(instancia.getProperty("campopesquisa"));
		vamb.setSleep(instancia.getProperty("sleep"));
		vamb.setBaudrate(instancia.getProperty("baudrate"));
		vamb.setSerialporta(instancia.getProperty("portaserial"));
```

```
            vamb.setModoentrada(instancia.getProperty("modoentrada"));
            vamb.setColunains1(instancia.getProperty("colunainsert1"));
            vamb.setColunareg1(instancia.getProperty("colunaregistro1"));
            vamb.setColunareg2(instancia.getProperty("colunaregistro2"));

            log.debug("Driver: " + vamb.getDriver());
            log.debug("Path: " + vamb.getPath());
            log.debug("BaudRate: " + vamb.getBaudrate());
            log.debug("SerialPorta: " + vamb.getSerialporta());
            log.debug("ModoEntrada: " + vamb.getModoentrada());
            log.debug("SleepTime: " + vamb.getSleep());
            log.debug("Tabela Consulta: " + vamb.getTabelacon());
            log.debug("Tabela Insert: " + vamb.getTabelains());
            log.debug("Tabela Registro: " + vamb.getTabelareg());
            log.debug("Campo da tabela: " + vamb.getCampo1tabela());
            log.debug("Campo da tabela: " + vamb.getCampo2tabela());
            log.debug("Dados1tabela: " + vamb.getDados1tabela());
            log.debug("Dados2tabela: " + vamb.getDados2tabela());
            log.debug("CampoPesquisa: " + vamb.getCampopesquisa());
            log.debug("ColunaInsert1: " + vamb.getColunains1());
            log.debug("ColunaRegistro1: " + vamb.getColunareg1());
            log.debug("ColunaRegistro1: " + vamb.getColunareg2());

            JOptionPane.showMessageDialog(null,
            "Carga de variáveis do ambiente realizada");

}

// Método principal, responável pela chamada dos demais métodos
public void start() throws Exception{

            log.setLevel(Level.toLevel(vamb.getNivel_log()));

            String entradadados = "";
            String resposta = "";
            boolean testetoken;
            StringTokenizer stoken = null;
            String registro = "";
            String escritaOk = null;

            vamb.setDados1tabela(null);
            vamb.setDados2tabela(null);

            do {

                        entradadados = null;
                        testetoken = true;
                        stoken = null;
                        String mode = "";
                        String controle = "";

                        mode = vamb.getModoentrada();

                        log.debug("Valor do modoentrada: " + mode);
                        JOptionPane.showMessageDialog(null,
                        "Definindo Modo entrada");


                        if (mode.equals("USB")){
```

```
		JOptionPane.showMessageDialog(null,
		"Modo USB");

		log.info("Aguardando input de dados...");

		// Aguarda entrada de dados
		do{
			JOptionPane.showMessageDialog(null,
			"Aguardando entrada de dados");
			entradadados = serial.usbreader();
			log.debug("Valor do entradados: " + entradadados);
			if (entradadados.equals("vazio")){

				controle = "True";

			}else{

				controle = "False";

			}
		} while (controle.equals("True"));
}

else if (mode.equals("SERIAL")){

		JOptionPane.showMessageDialog(null,
		"Modo Serial");

		log.info("Aguardando input de dados...");
		String comport = vamb.getSerialporta();
		int brate = Integer.parseInt(vamb.getBaudrate());
		int sleeptime = Integer.parseInt(vamb.getSleep());

		// Aguarda entrada de dados
		do{
			JOptionPane.showMessageDialog(null,
			"Aguardando entrada de dados");
			entradadados = serial.reader(comport, brate, sleeptime);
			log.debug("Valor do entradados: " + entradadados);
			if (entradadados.equals("vazio")){

				controle = "True";

			}else{

				controle = "False";

			}
		} while (controle.equals("True"));

} else {

			log.info("Modo de entrada inválido");
			log.debug("Valor do modoentrada: " + mode);
			break;

}

if (entradadados.equals("ERRO")){
```

```
                    testetoken = false;

                }else {

                        log.debug("Valor entradadados: " +entradadados);
                        JOptionPane.showMessageDialog(null,
                        "Dado recebido, iniciando validação");

                        stoken = new StringTokenizer(entradadados, ";");

                        int qtdtoken = stoken.countTokens();

                        // Verifica a estrutura da string recebida
                        testetoken = qtdtoken != 2;
                        log.debug("Testetoken é: " +entradadados);

                        if (testetoken == true) {

                                log.info("");
                                log.info("Input enviado esta errado!!!");
                                log.info("Contém mais de 2 parametros separados
por ;");
                                log.info("Deve ser enviado nova string");
                                log.info("");

                        }

                }

        } while (testetoken == true);

        if (entradadados != "ERRO"){

                        vamb.setDados1tabela(stoken.nextToken());
                        log.debug(vamb.getDados1tabela());
                        vamb.setDados2tabela(stoken.nextToken());
                        log.debug(vamb.getDados2tabela());

                        int operacao = 0;

                        JOptionPane.showMessageDialog(null,
                        "Dado válido, definindo operação");

                        // Define a operação recebida na serial
                        if (vamb.getDados2tabela().startsWith("INS")) {
                                operacao = 1;
                        } else if (vamb.getDados2tabela().startsWith("REG")) {
                                operacao = 2;
                        } else if (vamb.getDados2tabela().startsWith("CON")) {
                                operacao = 3;
                        }

                        switch (operacao) {

                        case 1:
                                log.info("");
                                log.info("Operação de Insert");
                                log.info("");

                                JOptionPane.showMessageDialog(null,
```

```
                                        "Operação insert iniciando");

                                        try{

                                                bo.AtualizaInsertBanco(vamb.getDriver(),
vamb.getPath(), vamb.getDados1tabela(), vamb.getTabelains(), vamb.getColunains1());

                                        } catch (Exception e){
                                                log.info(e.getMessage(), e);
                                                JOptionPane.showMessageDialog(null,
                                                "Erro: Operação insert não executada, verificar log
para detalhes");
                                        }

                                        JOptionPane.showMessageDialog(null,
                                        "Operação insert finalizada");

                                        break;

                                case 2:
                                        log.info("");
                                        log.info("Operação de Registro");
                                        log.info("");

                                        JOptionPane.showMessageDialog(null,
                                        "Operação registro iniciando");

                                        while (registro == null || registro.equals("")){

                                                registro = JOptionPane.showInputDialog ("Insira
registro adicional da Tag");
                                                if (registro == null || registro.equals("")){
                                                        JOptionPane.showMessageDialog(null,
                                                        "Deve ser inserido um registro para tag");
                                                        }
                                                }

                                        try{

                                                bo.AtualizaRegistroBanco(vamb.getDriver(),
vamb.getPath(), vamb.getDados1tabela(), registro, vamb.getTabelareg(), vamb.getColunareg1(),
vamb.getColunareg2());

                                        } catch (Exception e){
                                                log.info(e.getMessage(), e);
                                                JOptionPane.showMessageDialog(null,
                                                "Erro: Operação registro não executada, verificar log
para detalhes");
                                        }

                                        JOptionPane.showMessageDialog(null,
                                        "Operação registro finalizada");

                                        break;

                                case 3:
                                        log.info("");
                                        log.info("Operação de Consulta");
                                        log.info("");
```

```
                                        JOptionPane.showMessageDialog(null,
                                        "Operação consulta iniciando");

                                        Vector rs = bo.SelectBanco(vamb.getCampo1tabela(),
vamb.getCampo2tabela(), vamb.getTabelacon(), vamb.getCampopesquisa(), vamb.getDados1tabela(),
vamb.getDriver(), vamb.getPath());

                                        log.info("rs.size: " + rs.size());

                                        if (rs.size() > 0) {

                                                log.info("");
                                                log.info("Resultado da Consulta no banco:");
                                                log.info("DADO1: " + rs.get(0));
                                                log.info("DADO2: " + rs.get(1));
                                                log.info("");
                                                resposta = (String) rs.get(0);
                                                String comport = vamb.getSerialporta();
                                                int brate = Integer.parseInt(vamb.getBaudrate());
                                                int sleeptime = Integer.parseInt(vamb.getSleep());
                                                escritaOk = serial.writer(resposta, comport, brate,
sleeptime);
                                                if (escritaOk.equals("SUCESSO")){

                                                        log.info("Dados enviados");
                                                        JOptionPane.showMessageDialog(null,
                                                        "Dados Enviados");

                                                }else{

                                                        log.info("ERRO: Erro na escrita, consultar
log para detalhes");
                                                        JOptionPane.showMessageDialog(null,
                                                        "Erro na escrita, consultar log para
detalhes");

                                                }


                                        } else {

                                                log.info("");
                                                log.info("Dado consultado não existe no banco!!!");
                                                log.info("");
                                                JOptionPane.showMessageDialog(null,
                                                "Dado consultado não existe no banco, sem retorno
para serial");

                                        }

                                        JOptionPane.showMessageDialog(null,
                                        "Operação consulta finalizada");

                                        break;

                                default:
                                        log.info("Esta operação (" + vamb.getOperacao() + ") não é
válida!");

                                        JOptionPane.showMessageDialog(null,
```

```
                                        "Operação não prevista");

                                }
                }else{

                        log.info("ERRO: Erro nos processos da Serial, consulte log para detalhes");
                        JOptionPane.showMessageDialog(null,
                        "Erro nos processos da Serial, consulte log para detalhes");
                }

        }

        public static void main(String[] args) throws Exception {


                Rotina rot = new Rotina();

                try {

                        String lock = "SIM";

                        while (lock.equals("SIM")) {

                                rot.carga();
                                log.setLevel(Level.toLevel(vamb.getNivel_log()));
                                rot.start();

                                lock = null;

                                while (lock == null || lock.equals("")){

                                        lock = JOptionPane.showInputDialog ("Deseja Continuar?
SIM ou NAO?");
                                        if (lock == null || lock.equals("")){
                                                JOptionPane.showMessageDialog(null,
                                                "Deve ser informado SIM ou NAO!");
                                                }
                                        }

                        }

                } catch (Exception e) {
                        e.printStackTrace();
                        log.debug(e.getMessage(), e);
                }

                System.exit(0);

        }

}
```

Package BancodeDados

Classe BancoOperacoes

```
// Classe responsável pela conexão e operações no banco de dados

package BancodeDados;


import java.sql.Clob;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.Vector;

import javax.swing.JOptionPane;

import org.apache.log4j.Level;
import org.apache.log4j.Logger;

import Log.ConfigLog;

public class BancoOperacoes {

	private static final Logger log = ConfigLog.getLogger();
	private static ResultSet res = null;
	private static Statement stm = null;
	static Vector campos;
	static Vector dados;
	static Vector consulta;

	//Método que realiza conexão com o banco
	private java.sql.Connection getConnection(String Driver, String Path) throws Exception {

		log.setLevel(Level.toLevel("debug"));

		java.sql.Connection con = null;
		try {
			Class.forName(Driver);
			con = DriverManager.getConnection(Path);
		} catch (Exception e) {
			log.debug(e.getMessage(), e);
			JOptionPane.showMessageDialog(null,
			"Erro: Erro na conexão com o banco, , verificar log para detalhes");
			throw e;
		}
		return con;
	}
	// Método que realiza operação Insert com o banco
	public void AtualizaInsertBanco(String Driver, String Path, String Dados1, String tabelains,
String colunains1 ) {

		log.setLevel(Level.toLevel("debug"));

		java.sql.Connection conBD = null;
		PreparedStatement ps = null;
		try {
```

```
                        log.info("");
                        log.info("Setando variaveis para gravar no banco...");
                        log.info("");
                        log.info("Abrindo conexão...");
                        log.info("");

                        conBD = getConnection(Driver, Path);
                        log.debug("Driver" + Driver);
                        log.debug("Path" + Path);

                        log.info("");
                        log.info("populando dados do insert...");
                        log.info("");

                        String sql = "INSERT INTO "+tabelains +"("+colunains1+", DATA) VALUES
(? , SYSDATE)";
                        log.debug("SQL: " +sql);

                        ps = conBD.prepareStatement("INSERT INTO "+tabelains +"("+colunains1+",
DATA) VALUES (? , SYSDATE)");
                        ps.setString(1, Dados1);

                        log.info("Executando insert...");
                        log.info("");

                        ps.executeUpdate();

                } catch (Exception e) {
                        log.debug(e.getMessage(), e);
                        JOptionPane.showMessageDialog(null,
                        "Erro: Operação insert não executada, verificar log para detalhes");
                } finally {
                        try {
                                if (null != conBD)
                                        log.info("");
                                        log.info("Insert Executado com sucesso!");
                                        log.info("");
                                        log.info("Fechando conexão...");
                                        log.info("");
                                        conBD.close();

                        } catch (SQLException e) {
                                log.debug(e.getMessage(), e);
                                JOptionPane.showMessageDialog(null,
                                "Erro: Operação insert não executada, verificar log para detalhes");
                        }
                }
        }

        // Método que realiza operação Registro com o banco
        public void AtualizaRegistroBanco(String Driver, String Path, String Dados1, String Dados2,
String tabelareg, String colunareg1, String colunareg2){

                log.setLevel(Level.toLevel("debug"));

                java.sql.Connection conBD = null;
                PreparedStatement ps = null;
                try {

                        log.info("");
```

```
                        log.info("Setando variaveis para gravar no banco...");
                        log.info("");
                        log.info("Abrindo conexão...");
                        log.info("");

                        conBD = getConnection(Driver, Path);

                        log.info("");
                        log.info("populando dados do insert...");
                        log.info("");

                        String sql = "INSERT INTO "+ tabelareg + " ("+colunareg1+","+
colunareg2+", DATA) VALUES (? , ?, SYSDATE)";
                        log.debug("SQL: " +sql);

                        ps = conBD.prepareStatement("INSERT INTO "+ tabelareg + "
("+colunareg1+","+ colunareg2+", DATA) VALUES (? , ?, SYSDATE)");
                        ps.setString(1, Dados1);
                        ps.setString(2, Dados2);

                        log.info("Executando insert...");
                        log.info("");

                        ps.executeUpdate();

                } catch (Exception e) {
                        log.debug(e.getMessage(), e);
                        JOptionPane.showMessageDialog(null,
                        "Erro: Operação registro não executada, verificar log para detalhes");
                } finally {
                        try {
                                if (null != conBD)
                                        log.info("");
                                        log.info("Insert Executado com sucesso!");
                                        log.info("");
                                        log.info("Fechando conexão...");
                                        log.info("");
                                        conBD.close();

                        } catch (SQLException e) {
                                log.debug(e.getMessage(), e);
                                JOptionPane.showMessageDialog(null,
                                "Erro: Operação registro não executada, verificar log para detalhes");
                        }
                }
        }

        // Método que realiza operação Consulta com o banco
        public Vector SelectBanco(String Campo, String Campo2, String Tabela, String CampoPesquisa,
String Dado, String Driver, String Path) throws Exception {

                log.setLevel(Level.toLevel("debug"));

                java.sql.Connection conBD = null;
                PreparedStatement ps = null;
                consulta = new Vector();
                dados = new Vector();
                campos = new Vector();

                try {
```

```
                    log.info("");
                    log.info("Setando variaveis para fazer a consulta no banco...");
                    log.info("");
                    log.info("Abrindo conexão...");
                    log.info("");

                    conBD = getConnection(Driver, Path);

                    log.info("");
                    log.info("populando dados do select...");
                    log.info("");

                    String sql = "SELECT " + Campo + " , " + Campo2 + " FROM " + Tabela + "
WHERE " + CampoPesquisa + " = '" + Dado + "' ";
                    log.debug("sql: " + sql );

                    log.info("Executando select...");
                    log.info("");

                    stm = conBD.createStatement();
                    res = stm.executeQuery(sql);

                    for (int i = 0; i < res.getMetaData().getColumnCount(); i++) {
                            campos.add(i, res.getMetaData().getColumnLabel(i + 1));
                    }

                    consulta.add(0, campos);

                    while (res.next()) {
                            dados = new Vector();
                            for (int i = 0; i < campos.size(); i++) {
                                    if (!(res.getString(i + 1) == null)) {
                                            dados.add(i, res.getString(i + 1));
                                    } else {
                                            Clob clob = res.getClob(i + 1);
                                            int strAux = (int) clob.length();
                                            dados.add(i, clob.getSubString(1, strAux));
                                    }
                            }
                            consulta.add(res.getRow(), dados);
                    }

            log.debug("campos: " + campos.toString());
            log.debug("dados: " + dados.toString());

                    return dados;


            } catch (Exception e) {
                    log.debug(e.getMessage(), e);
                    JOptionPane.showMessageDialog(null,
                    "Erro: Operação consulta não executada, verificar log para detalhes");
                    throw e;
            } finally {
                    try {
                            if (null != conBD)
                                    log.info("");
                                    log.info("Select Executado com sucesso!");
                                    log.info("");
```

```
                                        log.info("Fechando conexão...");
                                        log.info("");
                                        stm.close();
                                        res.close();
                                        conBD.close();

                                } catch (SQLException e) {
                                        log.debug(e.getMessage(), e);
                                        JOptionPane.showMessageDialog(null,
                                        "Erro: Operação consulta não executada, verificar log para detalhes");
                                        throw e;
                                }
                        }
                }

}
```

Package Loader

Classe GetProperties

```
// Classe responsável por carregar os dados do arquivo properties para execução

package Loader;

import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.util.Properties;

import org.apache.log4j.Level;
import org.apache.log4j.Logger;


import Log.ConfigLog;

public class GetProperties {
        private static final String ARQUIVO_PROPRIEDADES = "middleware.properties";
        private static GetProperties instance;
        private static Properties properties;
        protected static Logger logg = ConfigLog.getLogger();

        // Método para criar instância
        public static GetProperties getInstance() throws Exception {

                if (instance == null) {

                        instance = new GetProperties();
                        instance.inicializarParametrosArquivo();
                }

                return instance;
        }

        // Método para inicializar os parâmetros do arquivo
        private void inicializarParametrosArquivo() throws Exception {
                logg.setLevel(Level.toLevel("error"));
                InputStream in = null;
                try {
```

```
                        in = new FileInputStream(new File(System.getProperty("user.dir")
                                        + System.getProperty("file.separator")
                                        + ARQUIVO_PROPRIEDADES));

                        properties = new Properties();
                        properties.load(in);

                } catch (Exception e) {
                        logg.error(e.getMessage(), e);
                        throw e;
                }
        }

        // Método que busca no arquivo properties as chaves e retorna os valores configurados
        public String getProperty(String chave) {
                logg.setLevel(Level.toLevel("debug"));
                String propriedade = properties.getProperty(chave);

                if (null == propriedade) {
                        propriedade = "";
                }
                logg.debug("Valor da variável propriedade: " +propriedade);
                return propriedade;
        }

}
```

Classe VariaveisAmbiente

```
// Classe responsável pela gestão das variáveis de ambiente

package Loader;

public class VariaveisAmbiente {

      // Configuração do banco
      private String ip;
      private String porta;
      private String sid;
      private String driver;
      private String path;

      // Acesso ao banco
      private String user;
      private String pswd;

      // Variáveis de insert no banco
      private String tabelains;
      private String tabelareg;
      private String dados1tabela;
      private String dados2tabela;
      private String colunains1;
      private String colunareg1;
      private String colunareg2;

      // Variáveis de consulta no banco
      private String tabelacon;
      private String campo1tabela;
      private String campo2tabela;
      private String campopesquisa;
```

```
// Dados da Serial
private String baudrate;
private String serialporta;
private String sleep;

// Variáveis de entrada na serial
private String tag;
private String operacao;
public String modoentrada;

// Configurações do ambiente
private String nivel_log;

// Métodos
public String getIp() {
	return ip;
}
public void setIp(String ip) {
	this.ip = ip;
}
public String getPorta() {
	return porta;
}
public void setPorta(String porta) {
	this.porta = porta;
}
public String getSid() {
	return sid;
}
public void setSid(String sid) {
	this.sid = sid;
}
public String getUser() {
	return user;
}
public void setUser(String user) {
	this.user = user;
}
public String getPswd() {
	return pswd;
}
public void setPswd(String pswd) {
	this.pswd = pswd;
}
public String getTag() {
	return tag;
}
public void setTag(String tag) {
	this.tag = tag;
}
public String getOperacao() {
	return operacao;
}
public void setOperacao(String operacao) {
	this.operacao = operacao;
}
public String getDriver() {
	return driver;
}
public void setDriver(String driver) {
```

```
	this.driver = driver;
}
public String getPath() {
	return path;
}
public void setPath(String path) {
	this.path = path;
}
public String getNivel_log() {
	return nivel_log;
}
public void setNivel_log(String nivel_log) {
	this.nivel_log = nivel_log;
}
public String getCampo1tabela() {
	return campo1tabela;
}
public void setCampo1tabela(String campo1tabela) {
	this.campo1tabela = campo1tabela;
}
public String getCampo2tabela() {
	return campo2tabela;
}
public void setCampo2tabela(String campo2tabela) {
	this.campo2tabela = campo2tabela;
}
public String getDados1tabela() {
	return dados1tabela;
}
public void setDados1tabela(String dados1tabela) {
	this.dados1tabela = dados1tabela;
}
public String getDados2tabela() {
	return dados2tabela;
}
public void setDados2tabela(String dados2tabela) {
	this.dados2tabela = dados2tabela;
}
public String getCampopesquisa() {
	return campopesquisa;
}
public void setCampopesquisa(String campopesquisa) {
	this.campopesquisa = campopesquisa;
}
public String getSleep() {
	return sleep;
}
public void setSleep(String sleep) {
	this.sleep = sleep;
}
public String getBaudrate() {
	return baudrate;
}
public void setBaudrate(String baudrate) {
	this.baudrate = baudrate;
}
public String getSerialporta() {
	return serialporta;
}
public void setSerialporta(String serialporta) {
	this.serialporta = serialporta;
```

```
	}
	public String getModoentrada() {
		return modoentrada;
	}
	public void setModoentrada(String modoentrada) {
		this.modoentrada = modoentrada;
	}
	public String getTabelains() {
		return tabelains;
	}
	public void setTabelains(String tabelains) {
		this.tabelains = tabelains;
	}
	public String getTabelareg() {
		return tabelareg;
	}
	public void setTabelareg(String tabelareg) {
		this.tabelareg = tabelareg;
	}
	public String getTabelacon() {
		return tabelacon;
	}
	public void setTabelacon(String tabelacon) {
		this.tabelacon = tabelacon;
	}
	public String getColunains1() {
		return colunains1;
	}
	public void setColunains1(String colunains1) {
		this.colunains1 = colunains1;
	}
	public String getColunareg1() {
		return colunareg1;
	}
	public void setColunareg1(String colunareg1) {
		this.colunareg1 = colunareg1;
	}
	public String getColunareg2() {
		return colunareg2;
	}
	public void setColunareg2(String colunareg2) {
		this.colunareg2 = colunareg2;
	}
}
```

Package Log

Classe ConfigLog

// Classe responsável pela captura dos logs

package Log;

import java.util.Enumeration;
import java.util.Hashtable;

import org.apache.log4j.Logger;
import org.apache.log4j.PropertyConfigurator;

public class ConfigLog {

```
	public static long loggerCount = 0;
	public static final String Caminho = "log4j.properties";

	static {

		PropertyConfigurator.configure(Caminho);

	}


	public static Logger getLogger() {
		return Logger.getLogger(Long.toBinaryString(loggerCount++));
	}

}
```

Package Serial

Classe SerialCom

// Classe baseada no artigo "Utilizando a API RXTX para manipulação da serial" do site www.devmedia.com.br

```
package Serial;

import gnu.io.CommPortIdentifier;
import java.util.Enumeration;

public class SerialCom {

	protected String[] portas;
	protected Enumeration listaDePortas;

	// Construtor da Classe
	public SerialCom(){

    listaDePortas = CommPortIdentifier.getPortIdentifiers();

	}

	// Método responsável por retornar as portas disponíveis
	public String[] ObterPortas(){

    return portas;

	}

	// Método responsável por armazenar a lista de portas disponíveis no desktop
	protected void ListarPortas(){

    int i = 0;

    portas = new String[10];

    while (listaDePortas.hasMoreElements()) {

      CommPortIdentifier ips =

      (CommPortIdentifier)listaDePortas.nextElement();
```

```
            portas[i] = ips.getName();

            i++;

        }

            }

            // Método que valida se a porta setada está apta para comunicação
            public boolean PortaExiste(String COMp){

        String temp;

        boolean e = false;

        while (listaDePortas.hasMoreElements()) {

            CommPortIdentifier ips = (CommPortIdentifier)listaDePortas.nextElement();

            temp = ips.getName();

            if (temp.equals(COMp)== true) {

            e = true;

            }

        }

        return e;

            }

}
```

Classe SerialComLeitura

```
// Classe baseada no artigo "Utilizando a API RXTX para manipulação da serial" do site www.devmedia.com.br

package Serial;

import gnu.io.CommPortIdentifier;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;

import org.apache.log4j.Level;
import org.apache.log4j.Logger;
import Log.ConfigLog;

// Implementação dos métodos abstratos Run e SerialPortEventListener
public class SerialComLeitura implements Runnable, SerialPortEventListener {

                    public String Dadoslidos;
                    public int nodeBytes;
                    private int baudrate;
```

```
private int timeout;
private CommPortIdentifier cp;
private SerialPort porta;
private OutputStream saida;
private InputStream entrada;
private Thread threadLeitura;
private boolean IDPortaOK;
private boolean PortaOK;
private boolean Leitura;
private boolean Escrita;
private boolean LerOK;
private boolean EnviaOK;
private boolean FechaOK;
private String Porta;
protected String buffer;

private static final Logger log = ConfigLog.getLogger();

// Método responsável por armazenar o buffer lido
public void setBuffer(String buffer){

        this.buffer = buffer;

}

// Método responsável por retornar o buffer lido
public String getBuffer(){

        return buffer;

}

// Construtor responsável por receber os parâmetros da porta serial
public SerialComLeitura( String p , int b , int t ){

        this.Porta = p;

        this.baudrate = b;

        this.timeout = t;

}

// Método responsável por habilitar o controle lógico de escrita
public void HabilitarEscrita(){

        Escrita = true;

        Leitura = false;

}

// Método responsável por habilitar o controle lógico de leitura
public void HabilitarLeitura(){

        Escrita = false;

        Leitura = true;

}
```

```
// Método responsável por obter o ID da porta
public boolean ObterIdDaPorta(){

	log.setLevel(Level.toLevel("debug"));

	try {

		cp = CommPortIdentifier.getPortIdentifier(Porta);

		if ( cp == null ) {

			log.info("Erro na porta");

			IDPortaOK = false;

		}

		IDPortaOK = true;

	} catch (Exception e) {

		log.debug("Erro obtendo ID da porta: " + e);

		IDPortaOK = false;

	}

	return IDPortaOK;

}

// Método responsável por abrir a comunicação com a porta
public boolean AbrirPorta(){

	log.setLevel(Level.toLevel("debug"));

	try {

		porta = (SerialPort)cp.open("SerialComLeitura", timeout);

		PortaOK = true;

		//Configura os parâmetros de hardware da porta

		porta.setSerialPortParams(baudrate,

				porta.DATABITS_8,

				porta.STOPBITS_1,

				porta.PARITY_NONE);

		porta.setFlowControlMode(SerialPort.FLOWCONTROL_NONE);

	}catch(Exception e){

		PortaOK = false;

		log.debug("Erro abrindo comunicação: " + e);
```

```
		}

		return PortaOK;

	}

	// Método responsável pela leitura de dados
	public boolean LerDados(){

		log.setLevel(Level.toLevel("debug"));

		log.debug("Valor entrada " + entrada);

		if (Escrita == false){

			try {

				entrada = porta.getInputStream();

			} catch (Exception e) {

				log.debug("Erro de stream: " + e);

				LerOK = false;

			}

			try {

				porta.addEventListener(this);
				entrada = porta.getInputStream();
				log.debug("Valor da entrada: " + entrada);

			} catch (Exception e) {

				log.debug("Erro de listener: " + e);

				LerOK = false;

			}

			porta.notifyOnDataAvailable(true);

			try {

				threadLeitura = new Thread(this);

				threadLeitura.start();

				run();
				entrada = porta.getInputStream();

				log.debug("Valor da entrada: " + entrada);

				LerOK = true;

			} catch (Exception e) {

				log.debug("Erro de Thread: " + e);
```

```
                    LerOK = false;
                }
            }
            return LerOK;
        }
        // Método responsável pelo envio de dados
        public boolean EnviarUmaString(String msg){
            log.setLevel(Level.toLevel("debug"));
            if (Escrita==true) {
                try {
                    saida = porta.getOutputStream();
                    log.info("FLUXO OK!");
                } catch (Exception e) {
                    log.debug("Erro.STATUS: " + e );
                }
                try {
                    log.info("Enviando um byte para " + Porta );
                    log.debug("Enviando : " + msg );
                    saida.write(msg.getBytes());
                    Thread.sleep(100);
                    saida.flush();
                    EnviaOK = true;
                } catch (Exception e) {
                    log.info("Houve um erro durante o envio. ");
                    log.debug("STATUS: " + e );
                    EnviaOK = false;
                }
            } else {
                EnviaOK = false;
            }
            return EnviaOK;
```

```
        }

        // Método responsável pelo controle da thread de leitura
        public void run(){

                try {

                        Thread.sleep(5);

                } catch (Exception e) {

                        log.debug("Erro de Thred: " + e);

                }

        }

        // Método responsável pelo gerenciamento de todos os dados recebidos na serial
        public void serialEvent(SerialPortEvent ev){

                log.setLevel(Level.toLevel("debug"));

                StringBuffer bufferLeitura = new StringBuffer();
                log.debug("Valor do bufferLeitura: " + bufferLeitura);

                int novoDado = 0;

switch (ev.getEventType()) {

  case SerialPortEvent.BI:

  case SerialPortEvent.OE:

  case SerialPortEvent.FE:

  case SerialPortEvent.PE:

  case SerialPortEvent.CD:

  case SerialPortEvent.CTS:

  case SerialPortEvent.DSR:

  case SerialPortEvent.RI:

  case SerialPortEvent.OUTPUT_BUFFER_EMPTY:

        break;

  case SerialPortEvent.DATA_AVAILABLE:

        while(novoDado != -1){

                try{

                        novoDado = entrada.read();
                        log.debug("Valor do novoDado: " + novoDado);

                        if(novoDado == -1){
```

```
                                        break;

                                }

                                // Verifica se foi enviado o fim da string
                                if('\r' == (char)novoDado){

                                        bufferLeitura.append('\n');
                                        log.debug("Valor do novoDado: " + novoDado);

                                }else{

                                        bufferLeitura.append((char)novoDado);
                                        log.debug("Valor do novoDado: " + novoDado);

                                }

                        }catch(IOException ioe){

                                log.debug("Erro de leitura serial: " + ioe);

                        }

                }

                setBuffer(new String(bufferLeitura));

                log.debug("Valor do buffer: " + getBuffer());

                break;

        }

                }

                // Método responsável por fechar a porta
                public boolean FecharCom(){

                        log.setLevel(Level.toLevel("debug"));

                        try {

                                porta.close();
                                FechaOK = true;

                        } catch (Exception e) {

                                log.debug("Erro fechando porta: " + e);

                                FechaOK = false;

                        }

                        return FechaOK;

                }

                // Método responsável por retornar a porta
                public String obterPorta(){
```

```
		return Porta;

	}

	// Método responsável por retornar a baundrate
	public int obterBaudrate(){

		return baudrate;

	}

}
```

Classe Serial Connector

```
//Classe responsável pela montagem dos métodos Reader e Writter

package Serial;

import org.apache.log4j.Level;
import org.apache.log4j.Logger;

//import Loader.GetProperties;
//import Loader.VariaveisAmbiente;
import Log.ConfigLog;

public class SerialConnector {

	private static final Logger log = ConfigLog.getLogger();
	boolean controle = true;

	public String reader(String ComPort, int brate, int tout) {

		SerialComLeitura serial = new SerialComLeitura(ComPort, brate, tout);

		log.setLevel(Level.toLevel("debug"));

		String buffer = null;

		log.info("Processo Leitura inicio");

		serial.HabilitarLeitura();
		log.info("Leitura habilitada");

		controle = serial.ObterIdDaPorta();
		if (controle == true){

			log.info("Id porta obtido");

			controle = serial.AbrirPorta();
			if(controle == true){

				log.info("Porta aberta");

				controle = serial.LerDados();
				if (controle == true){

					log.info("Poolling de leitura iniciado");
```

```
                                        // Controle do tempo que o poolling mantém a porta da serial
aberta
                                try {

                                        Thread.sleep(tout);

                                    } catch (InterruptedException ex) {

                                        log.info("Erro na tread de leitura");
                                        log.debug("Erro na Thread: " + ex);

                                    }

                                    log.info("Poolling de leitura finalizado");

                                    }else{

                                        log.info("Erro ao Ler Porta");
                                        buffer = "ERRO";

                                    }

                            }else{

                                    log.info("Erro ao abrir Porta");
                                    buffer = "ERRO";

                            }

                    }else{

                            log.info("Erro ao obter ID da porta");
                            buffer = "ERRO";

                    }

            controle = serial.FecharCom();
            if (controle == true){

                    log.info("Leitura finalizada");
                    buffer = serial.getBuffer();

            }else{

                    log.info("Erro ao fechar Porta");
                            buffer = "ERRO";

            }

                    log.debug("Valor do buffer: " + buffer);
                    log.info("<>");

                    if (buffer==null){

                            buffer = "vazio";

                    }

                    return buffer;
            }
```

```
public String writer(String resposta, String ComPort, int brate, int tout) {

        SerialComLeitura serial = new SerialComLeitura(ComPort, brate, tout);
        String estado;

        log.setLevel(Level.toLevel("debug"));

   log.info("Escrita iniciada");

   serial.HabilitarEscrita();
        log.info("Escrita habilitada");

        controle = serial.ObterIdDaPorta();
        if (controle == true){

                log.info("Id porta obtido");

                controle = serial.AbrirPorta();
                if(controle == true){

                        log.info("Porta aberta");

                        controle = serial.EnviarUmaString(resposta);
                        if (controle == true){

                                log.info("Dados enviados");

                        }else{

                                log.info("Enviar Dados");
                                estado = "ERRO";

                        }

                }else{

                        log.info("Erro ao abrir Porta");
                        estado = "ERRO";

                }

        }else{

                log.info("Erro ao obter ID da porta");
                estado = "ERRO";

        }

        controle = serial.FecharCom();
if (controle == true){

        log.info("Porta fechada");
        estado = "SUCESSO";

}else{

        log.info("Erro ao fechar Porta");
                estado = "ERRO";
```

```
	}

		log.info("Escrita finalizada");

		return estado;
	}

	public String usbreader() {

		log.setLevel(Level.toLevel("debug"));

		String buffer = null;

		byte[] bytes = new byte[2048];
		try {
			System.in.read(bytes);
		} catch (Exception e) {
			log.debug("Erro: " + e);
		}
		char[] caracts = new char[160];
		for (int i = 0; i < 160; i++) {
			caracts[i] = (char) bytes[i];
		}
		String str = (new String(caracts)).trim();
		log.debug("Valor do str: " + str);
		buffer = str;

		if (buffer.equals("")){

			buffer = "vazio";

		}

		return buffer;
	}

}
```

APÊNDICE B – Evidências Caso 1

Dados da Massa

Tag: 123456789

Dados Configuração BD

String conexão banco: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE
Tabela de Insert: PRATELEIRA_1
Coluna de Insert: TAG

Dados Configuração Serial

Porta: COM1
Baudrate: 9600
Sleep: 10 segundos

Cenário: Enviar tag na porta serial com a operação INS

Passo 1 - Parametrizar arquivo middleware.properties

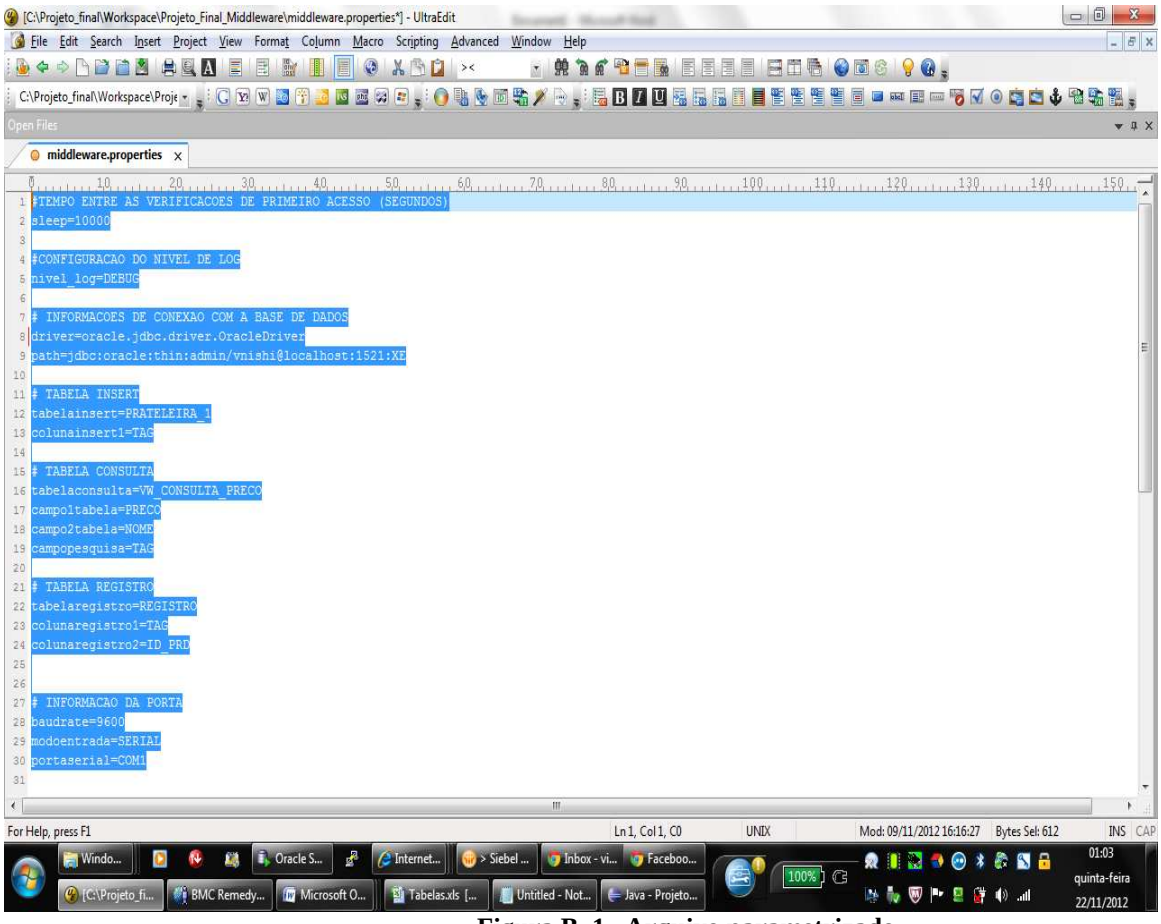

Figura B. 1– Arquivo parametrizado
Fonte: Autor

Passo 2 - Validar que dado não existe no banco

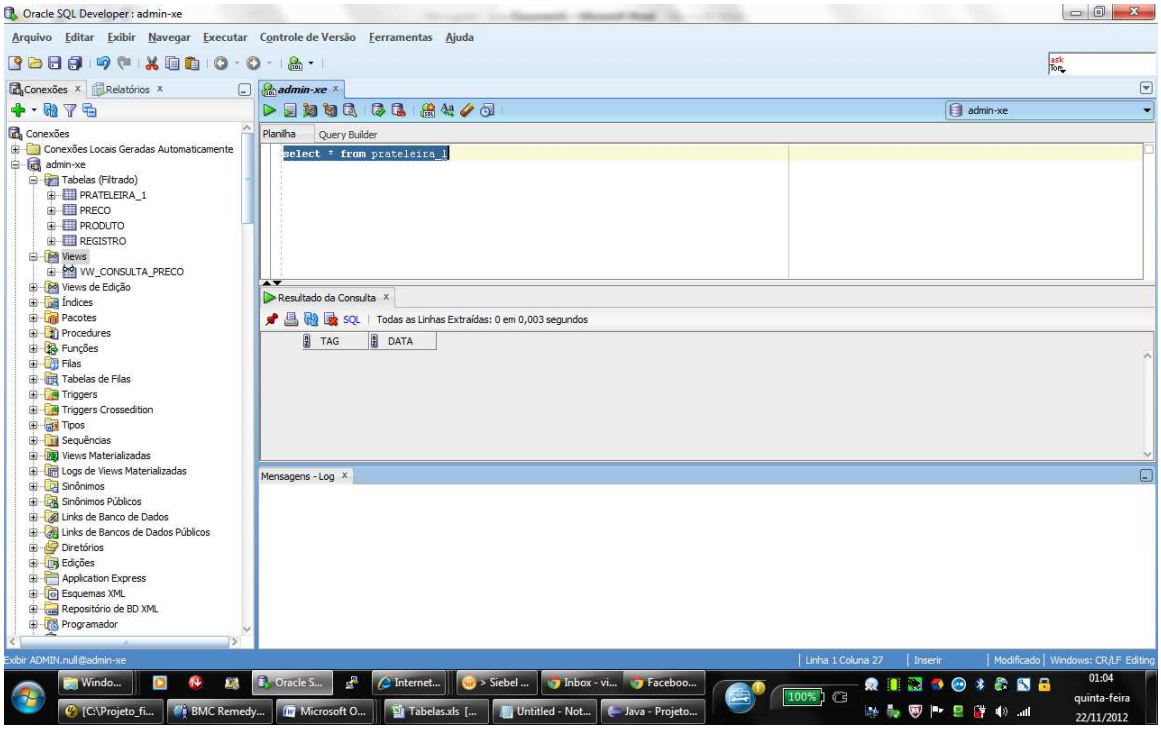

Figura B. 2– Validação
Fonte: Autor

Passo 3 - Iniciar Middleware;

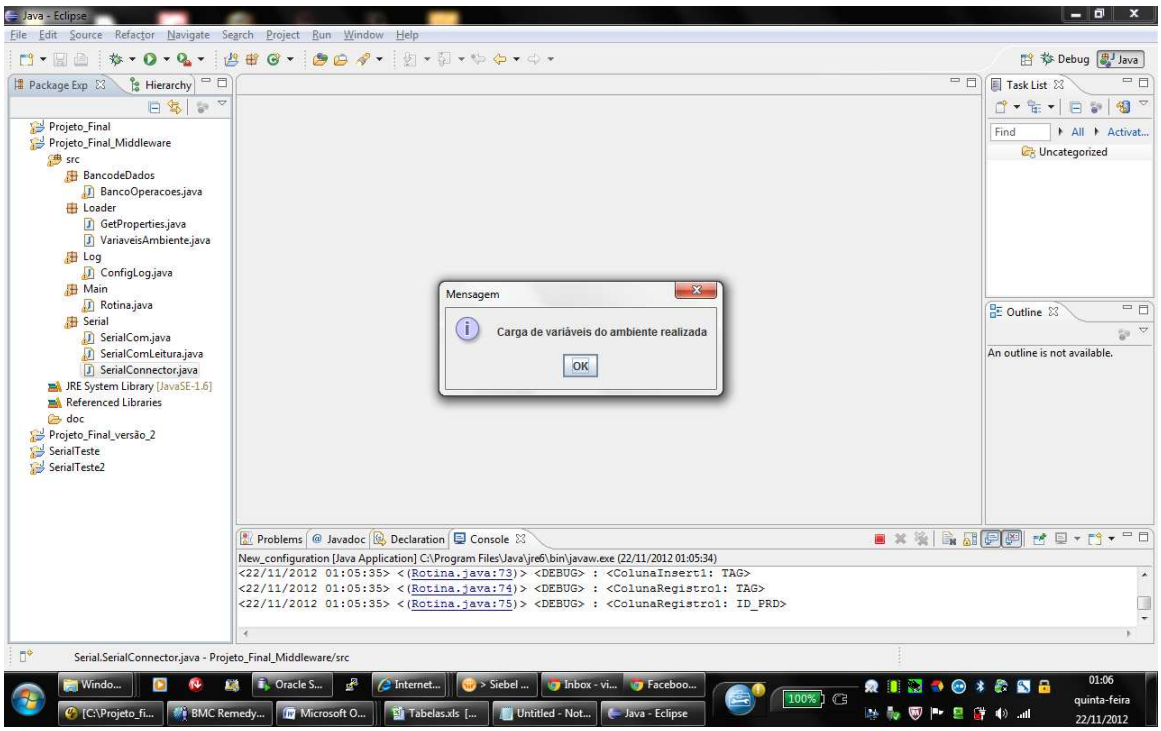

Figura B. 3– Inicio Carga
Fonte: Autor

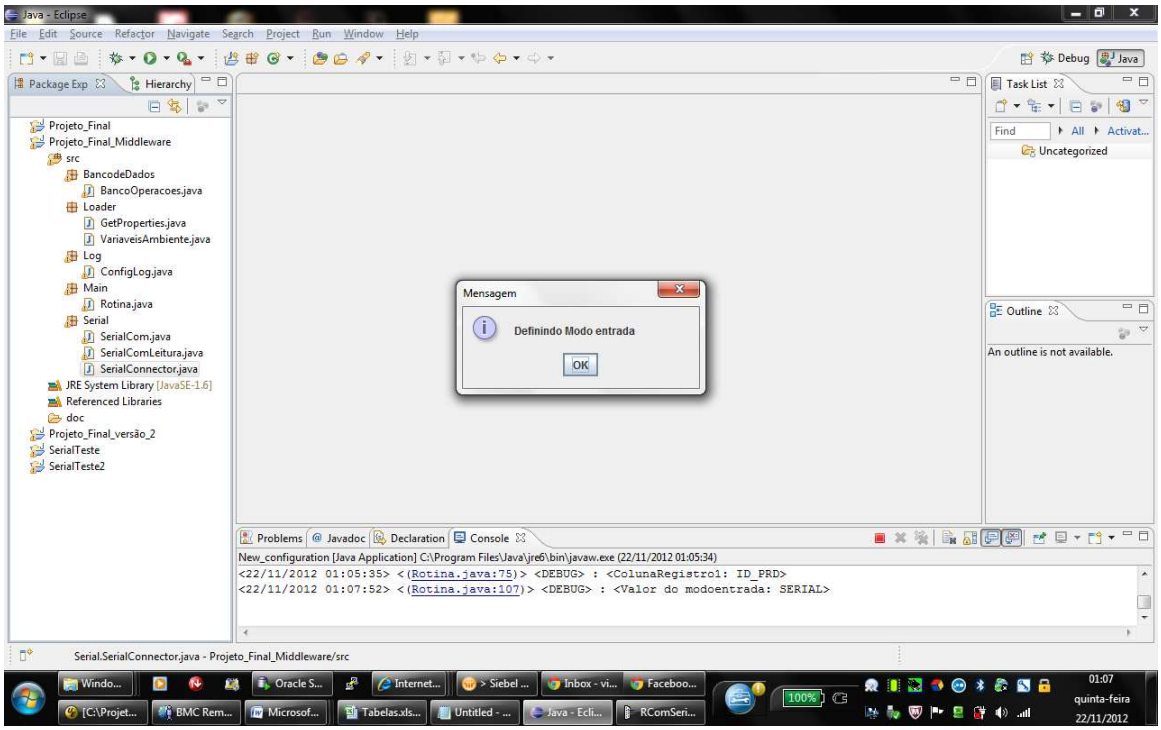

Figura B. 4– Modo Entrada
Fonte: Autor

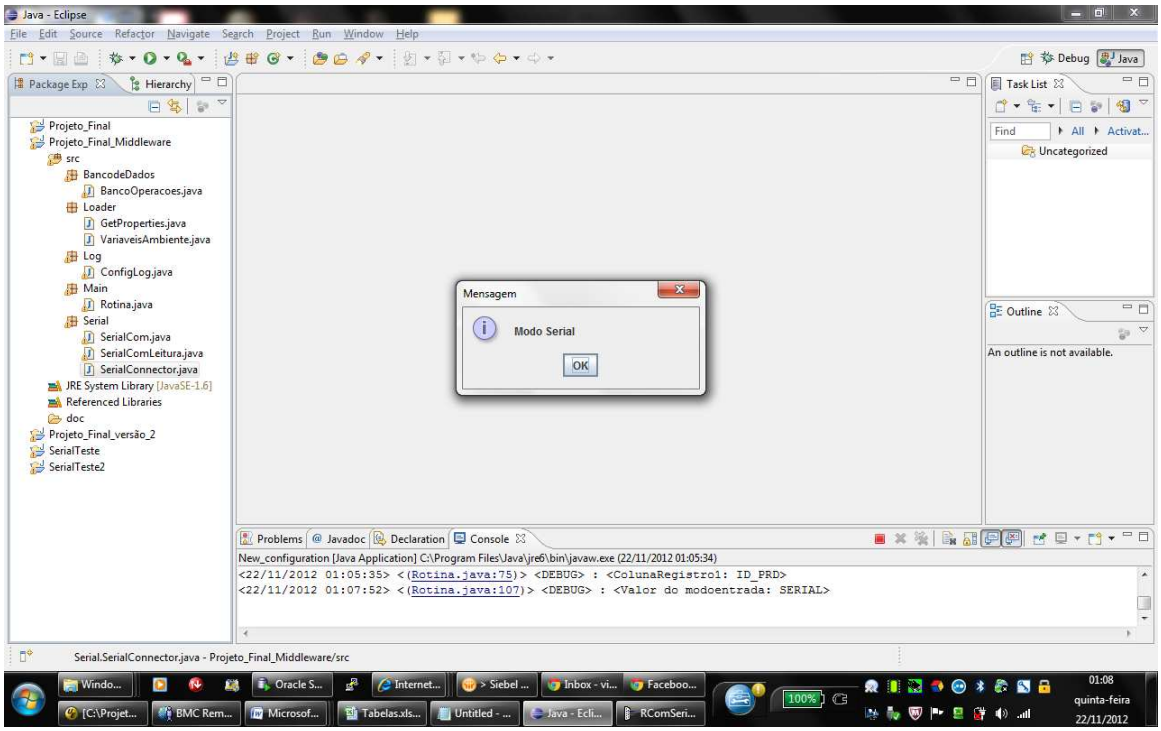

Figura B. 5– Modo Serial
Fonte: Autor

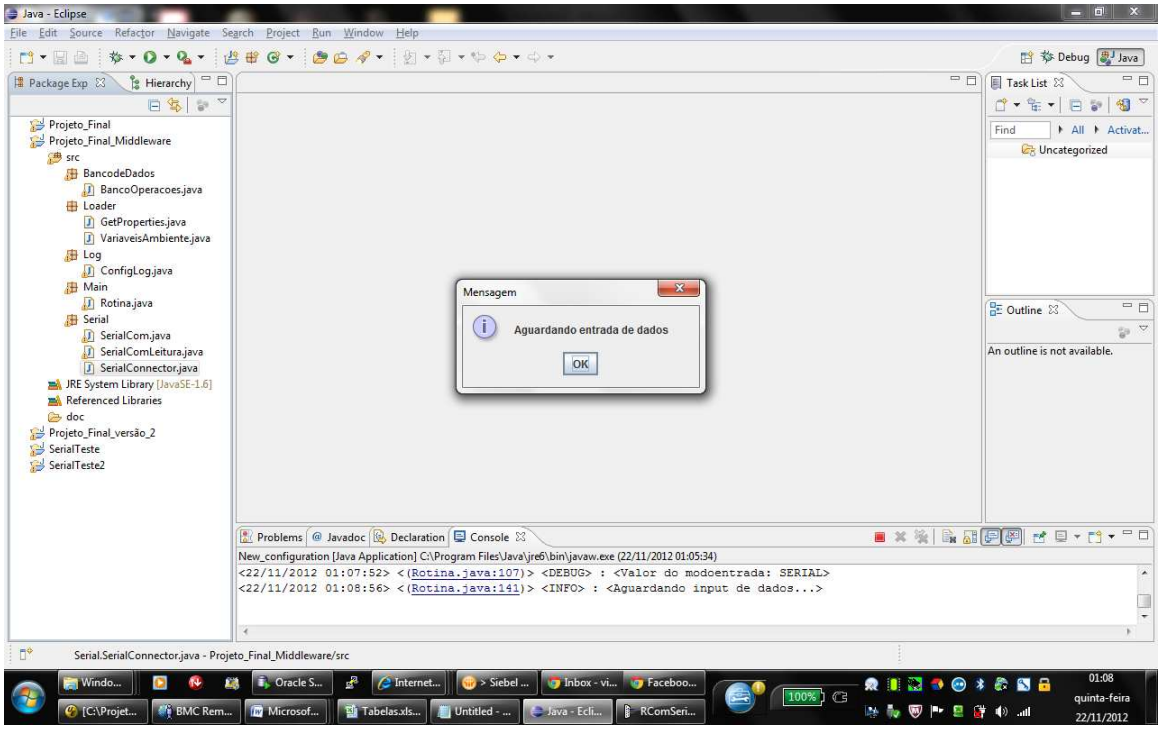

Figura B. 6– Aguardando dados
Fonte: Autor

Passo 4 - Enviar dado;

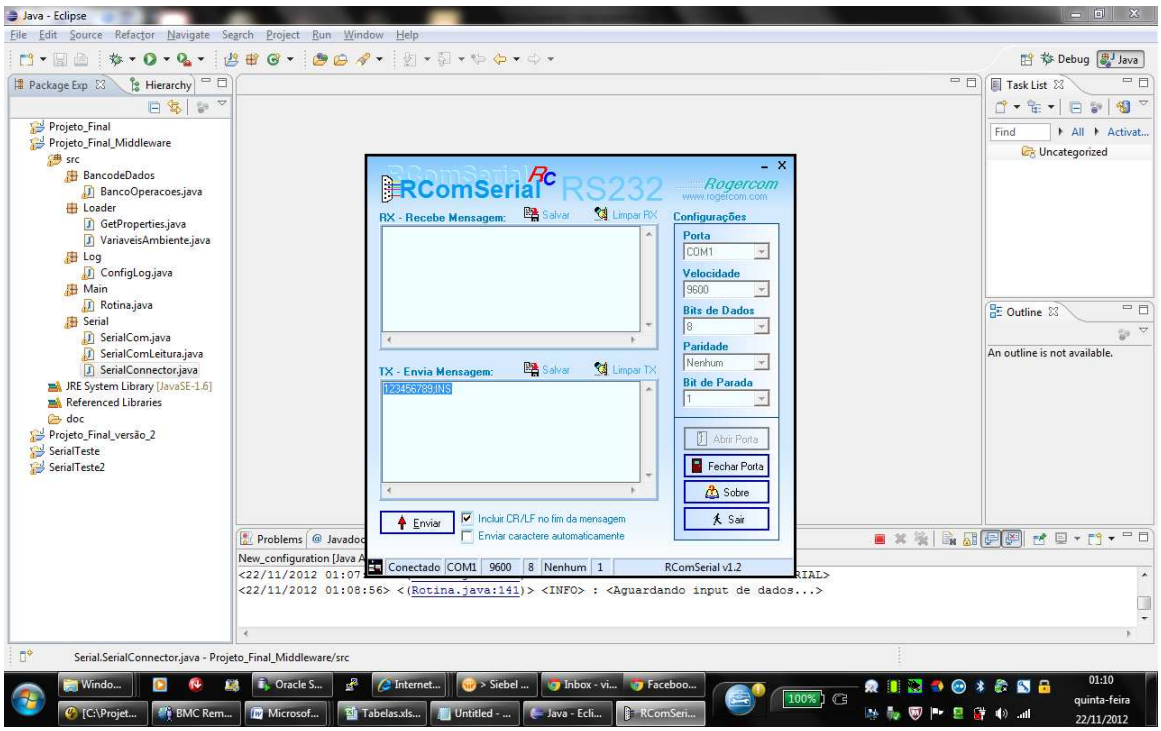

Figura B. 7– Enviar Dados
Fonte: Autor

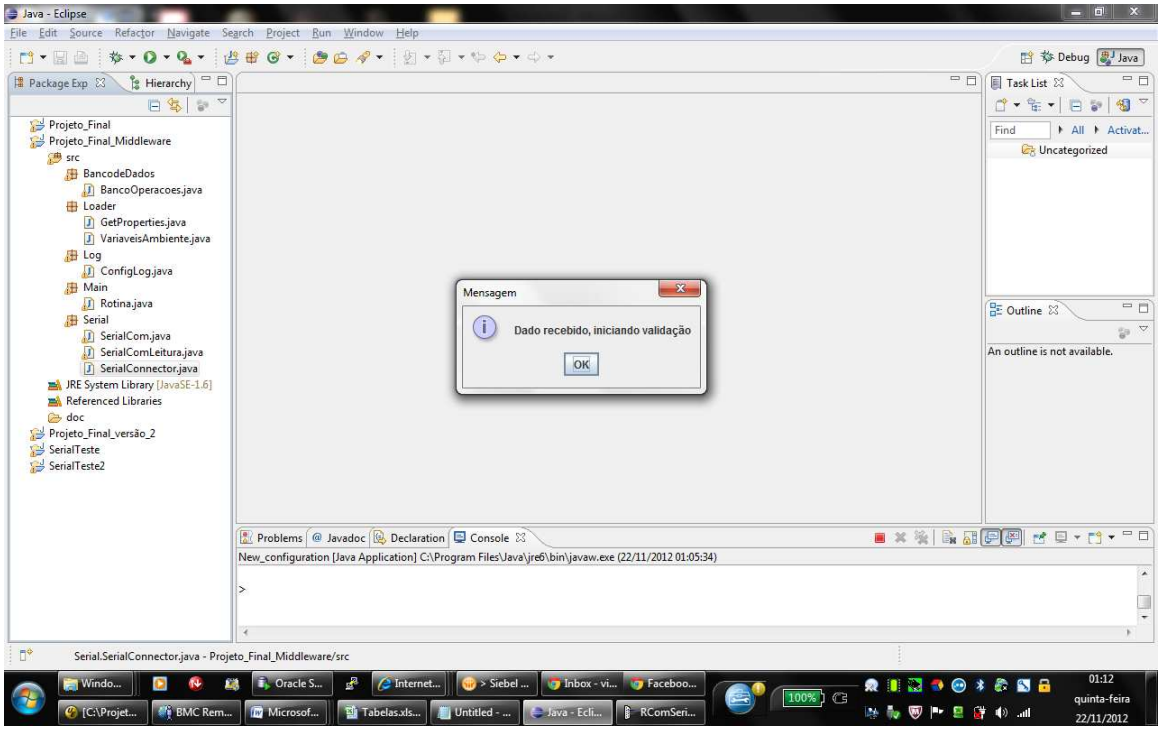

Figura B. 8– Dados Recebidos
Fonte: Autor

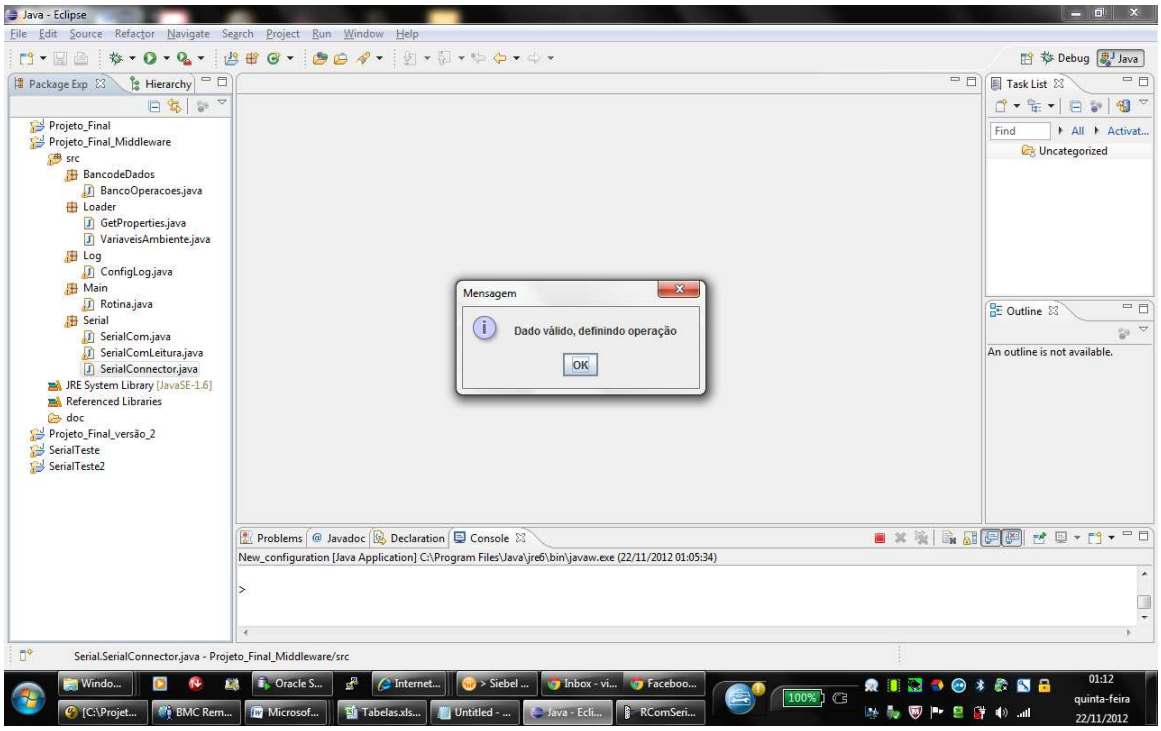

Figura B. 9– Dado Válido
Fonte: Autor

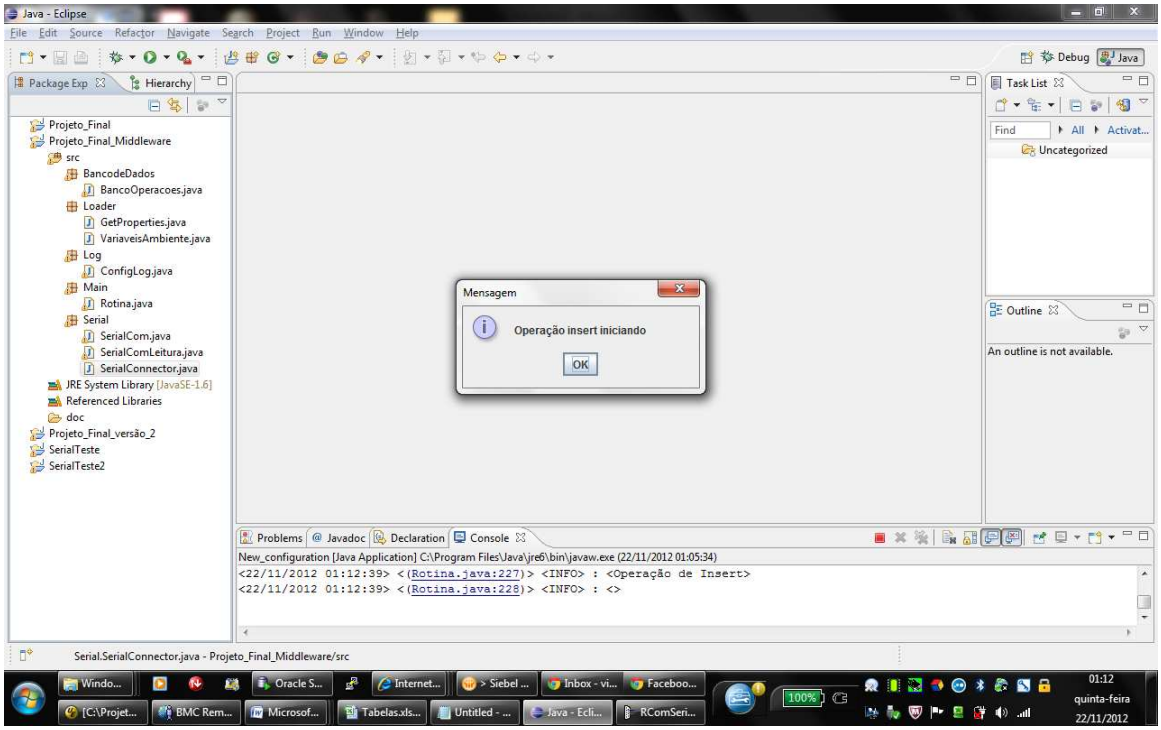

Figura B. 10– Operação Insert
Fonte: Autor

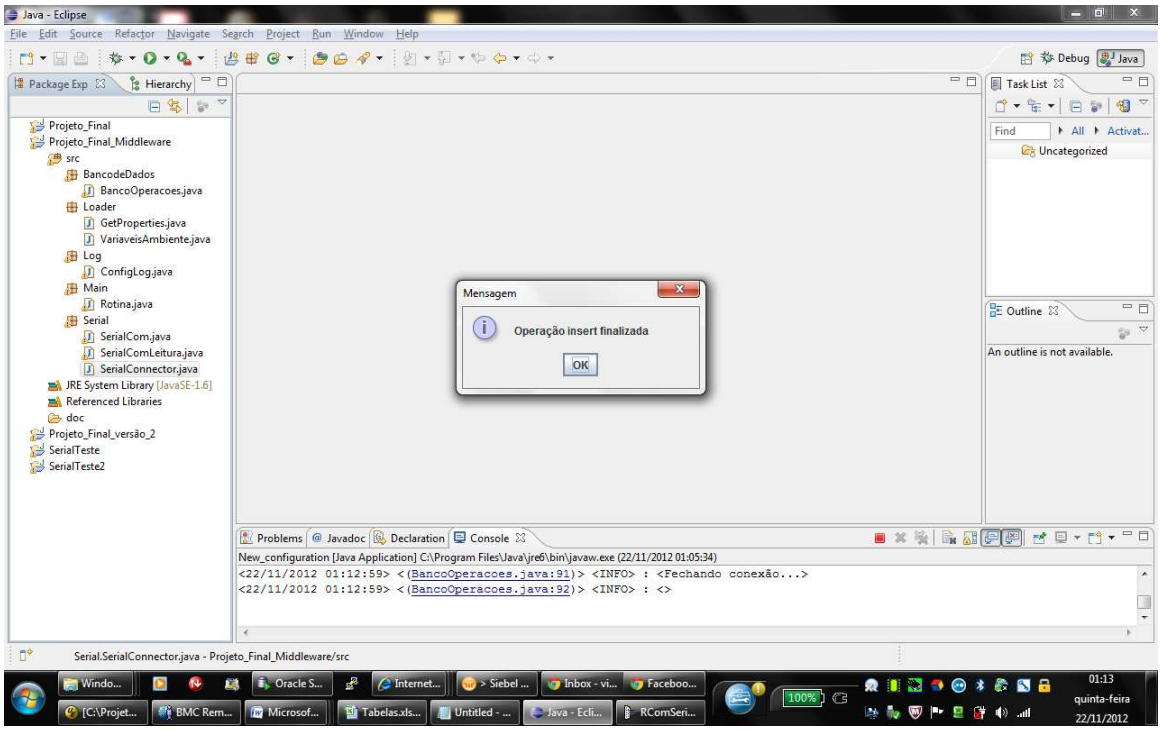

Figura B. 11– Final Operação
Fonte: Autor

Passo 5 - Pesquisar dados no banco.

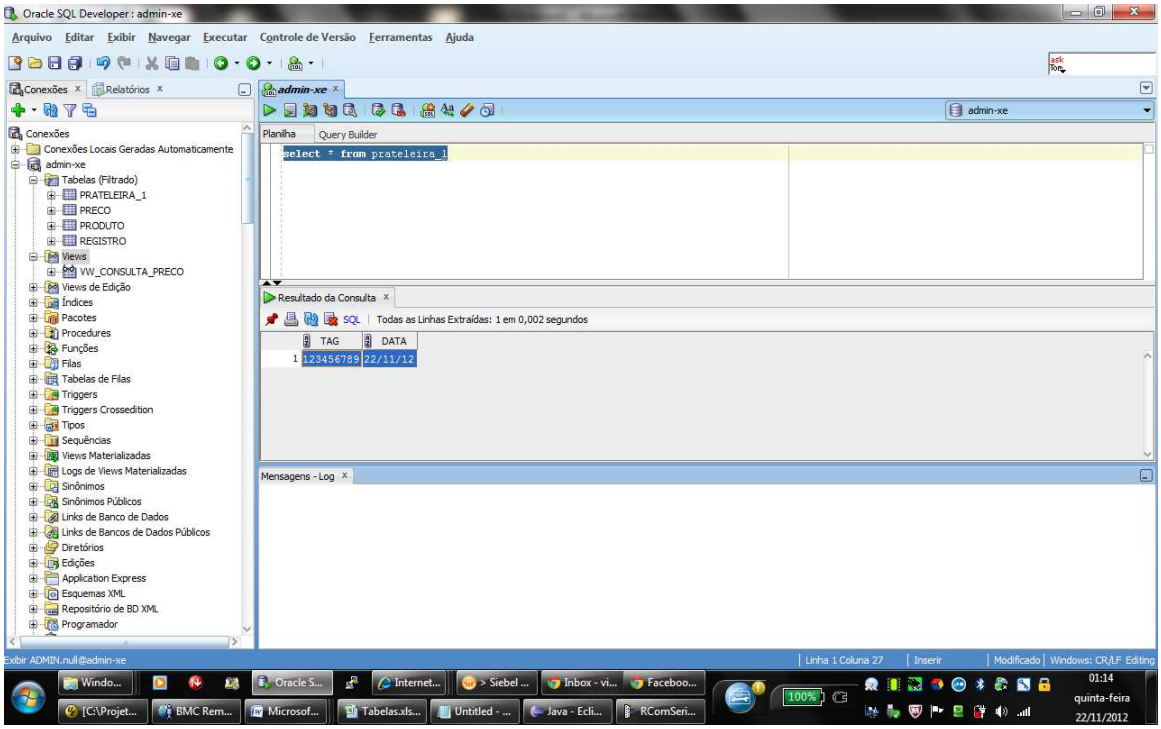

Figura B. 12– Pesquisar
Fonte: Autor

Log complento do cenário:

<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: DEBUG>
<22/11/2012 01:05:35> <(Rotina.java:36)> <INFO> : <Nível de LOG setado para: DEBUG>
<22/11/2012 01:05:35> <(Rotina.java:37)> <INFO> : <>
<22/11/2012 01:05:35> <(Rotina.java:39)> <INFO> : <>
<22/11/2012 01:05:35> <(Rotina.java:40)> <INFO> : <Setando variáveis iniciais do módulo...>
<22/11/2012 01:05:35> <(Rotina.java:41)> <INFO> : <>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: oracle.jdbc.driver.OracleDriver>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: VW_CONSULTA_PRECO>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRATELEIRA_1>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: REGISTRO>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRECO>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: NOME>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 10000>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 9600>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: COM1>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: SERIAL>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>

<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:05:35> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: ID_PRD>
<22/11/2012 01:05:35> <(Rotina.java:59)> <DEBUG> : <Driver: oracle.jdbc.driver.OracleDriver>
<22/11/2012 01:05:35> <(Rotina.java:60)> <DEBUG> : <Path: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 01:05:35> <(Rotina.java:61)> <DEBUG> : <BaudRate: 9600>
<22/11/2012 01:05:35> <(Rotina.java:62)> <DEBUG> : <SerialPorta: COM1>
<22/11/2012 01:05:35> <(Rotina.java:63)> <DEBUG> : <ModoEntrada: SERIAL>
<22/11/2012 01:05:35> <(Rotina.java:64)> <DEBUG> : <SleepTime: 10000>
<22/11/2012 01:05:35> <(Rotina.java:65)> <DEBUG> : <Tabela Consulta: VW_CONSULTA_PRECO>
<22/11/2012 01:05:35> <(Rotina.java:66)> <DEBUG> : <Tabela Insert: PRATELEIRA_1>
<22/11/2012 01:05:35> <(Rotina.java:67)> <DEBUG> : <Tabela Registro: REGISTRO>
<22/11/2012 01:05:35> <(Rotina.java:68)> <DEBUG> : <Campo da tabela: PRECO>
<22/11/2012 01:05:35> <(Rotina.java:69)> <DEBUG> : <Campo da tabela: NOME>
<22/11/2012 01:05:35> <(Rotina.java:70)> <DEBUG> : <Dados1tabela: null>
<22/11/2012 01:05:35> <(Rotina.java:71)> <DEBUG> : <Dados2tabela: null>
<22/11/2012 01:05:35> <(Rotina.java:72)> <DEBUG> : <CampoPesquisa: TAG>
<22/11/2012 01:05:35> <(Rotina.java:73)> <DEBUG> : <ColunaInsert1: TAG>
<22/11/2012 01:05:35> <(Rotina.java:74)> <DEBUG> : <ColunaRegistro1: TAG>
<22/11/2012 01:05:35> <(Rotina.java:75)> <DEBUG> : <ColunaRegistro1: ID_PRD>
<22/11/2012 01:07:52> <(Rotina.java:107)> <DEBUG> : <Valor do modoentrada: SERIAL>
<22/11/2012 01:08:56> <(Rotina.java:141)> <INFO> : <Aguardando input de dados...>
<22/11/2012 01:10:53> <(SerialConnector.java:25)> <INFO> : <Processo Leitura inicio>
<22/11/2012 01:10:53> <(SerialConnector.java:28)> <INFO> : <Leitura habilitada>
<22/11/2012 01:10:54> <(SerialConnector.java:33)> <INFO> : <Id porta obtido>
<22/11/2012 01:10:54> <(SerialConnector.java:38)> <INFO> : <Porta aberta>
<22/11/2012 01:10:54> <(SerialComLeitura.java:155)> <DEBUG> : <Valor entrada null>
<22/11/2012 01:10:54> <(SerialComLeitura.java:175)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@13caecd>
<22/11/2012 01:10:54> <(SerialComLeitura.java:196)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@13caecd>
<22/11/2012 01:10:54> <(SerialConnector.java:43)> <INFO> : <Poolling de leitura iniciado>
<22/11/2012 01:11:04> <(SerialConnector.java:57)> <INFO> : <Poolling de leitura finalizado>
<22/11/2012 01:11:04> <(SerialConnector.java:83)> <INFO> : <Leitura finalizada>
<22/11/2012 01:11:04> <(SerialConnector.java:93)> <DEBUG> : <Valor do buffer: null>
<22/11/2012 01:11:04> <(SerialConnector.java:94)> <INFO> : <<>>
<22/11/2012 01:11:04> <(Rotina.java:151)> <DEBUG> : <Valor do entradados: vazio>
<22/11/2012 01:11:49> <(SerialConnector.java:25)> <INFO> : <Processo Leitura inicio>
<22/11/2012 01:11:49> <(SerialConnector.java:28)> <INFO> : <Leitura habilitada>
<22/11/2012 01:11:49> <(SerialConnector.java:33)> <INFO> : <Id porta obtido>
<22/11/2012 01:11:49> <(SerialConnector.java:38)> <INFO> : <Porta aberta>
<22/11/2012 01:11:49> <(SerialComLeitura.java:155)> <DEBUG> : <Valor entrada null>
<22/11/2012 01:11:49> <(SerialComLeitura.java:175)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@64dc11>
<22/11/2012 01:11:49> <(SerialComLeitura.java:196)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@64dc11>
<22/11/2012 01:11:49> <(SerialConnector.java:43)> <INFO> : <Poolling de leitura iniciado>
<22/11/2012 01:11:51> <(SerialComLeitura.java:286)> <DEBUG> : <Valor do bufferLeitura: >
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 49>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 49>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 50>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 50>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 51>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 51>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 52>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 52>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 53>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 53>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 54>

<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 54>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 55>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 55>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 56>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 56>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 57>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 57>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 59>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 59>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 73>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 73>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 78>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 78>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 83>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 83>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 13>
<22/11/2012 01:11:51> <(SerialComLeitura.java:331)> <DEBUG> : <Valor do novoDado: 13>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 10>
<22/11/2012 01:11:51> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 10>
<22/11/2012 01:11:51> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: -1>
<22/11/2012 01:11:51> <(SerialComLeitura.java:350)> <DEBUG> : <Valor do buffer: 123456789;INS

>
<22/11/2012 01:11:59> <(SerialConnector.java:57)> <INFO> : <Poolling de leitura finalizado>
<22/11/2012 01:11:59> <(SerialConnector.java:83)> <INFO> : <Leitura finalizada>
<22/11/2012 01:11:59> <(SerialConnector.java:93)> <DEBUG> : <Valor do buffer: 123456789;INS

>
<22/11/2012 01:11:59> <(SerialConnector.java:94)> <INFO> : <<>>
<22/11/2012 01:11:59> <(Rotina.java:151)> <DEBUG> : <Valor do entradados: 123456789;INS

>
<22/11/2012 01:11:59> <(Rotina.java:176)> <DEBUG> : <Valor entradadados: 123456789;INS

>
<22/11/2012 01:12:22> <(Rotina.java:186)> <DEBUG> : <Testetoken é: 123456789;INS

>
<22/11/2012 01:12:22> <(Rotina.java:205)> <DEBUG> : <123456789>
<22/11/2012 01:12:22> <(Rotina.java:207)> <DEBUG> : <INS

>
<22/11/2012 01:12:39> <(Rotina.java:226)> <INFO> : <>
<22/11/2012 01:12:39> <(Rotina.java:227)> <INFO> : <Operação de Insert>
<22/11/2012 01:12:39> <(Rotina.java:228)> <INFO> : <>
<22/11/2012 01:12:57> <(BancoOperacoes.java:56)> <INFO> : <>
<22/11/2012 01:12:57> <(BancoOperacoes.java:57)> <INFO> : <Setando variaveis para gravar no banco...>
<22/11/2012 01:12:57> <(BancoOperacoes.java:58)> <INFO> : <>
<22/11/2012 01:12:57> <(BancoOperacoes.java:59)> <INFO> : <Abrindo conexão...>
<22/11/2012 01:12:57> <(BancoOperacoes.java:60)> <INFO> : <>
<22/11/2012 01:12:58> <(BancoOperacoes.java:63)> <DEBUG> : <Driveroracle.jdbc.driver.OracleDriver>
<22/11/2012 01:12:58> <(BancoOperacoes.java:64)> <DEBUG> : <Pathjdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 01:12:58> <(BancoOperacoes.java:66)> <INFO> : <>
<22/11/2012 01:12:58> <(BancoOperacoes.java:67)> <INFO> : <populando dados do insert...>
<22/11/2012 01:12:58> <(BancoOperacoes.java:68)> <INFO> : <>
<22/11/2012 01:12:58> <(BancoOperacoes.java:71)> <DEBUG> : <SQL: INSERT INTO PRATELEIRA_1(TAG, DATA) VALUES (? , SYSDATE)>

<22/11/2012 01:12:58> <(BancoOperacoes.java:76)> <INFO> : <Executando insert...>
<22/11/2012 01:12:58> <(BancoOperacoes.java:77)> <INFO> : <>
<22/11/2012 01:12:59> <(BancoOperacoes.java:88)> <INFO> : <>
<22/11/2012 01:12:59> <(BancoOperacoes.java:89)> <INFO> : <Insert Executado com sucesso!>
<22/11/2012 01:12:59> <(BancoOperacoes.java:90)> <INFO> : <>
<22/11/2012 01:12:59> <(BancoOperacoes.java:91)> <INFO> : <Fechando conexão...>
<22/11/2012 01:12:59> <(BancoOperacoes.java:92)> <INFO> : <>

APÊNDICE C – Evidências Caso 2

Dados da Massa

Tag: 987654321

Dados Configuração BD

String conexão banco: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE
Tabela de Insert: PRATELEIRA_1
Coluna de Insert: TAG

Dados Configuração Serial

Porta: COM1
Baudrate: 9600
Sleep: 10 segundos

Cenário: Enviar tag na porta USB com a operação INS

Passo 1 - Parametrizar arquivo middleware.properties

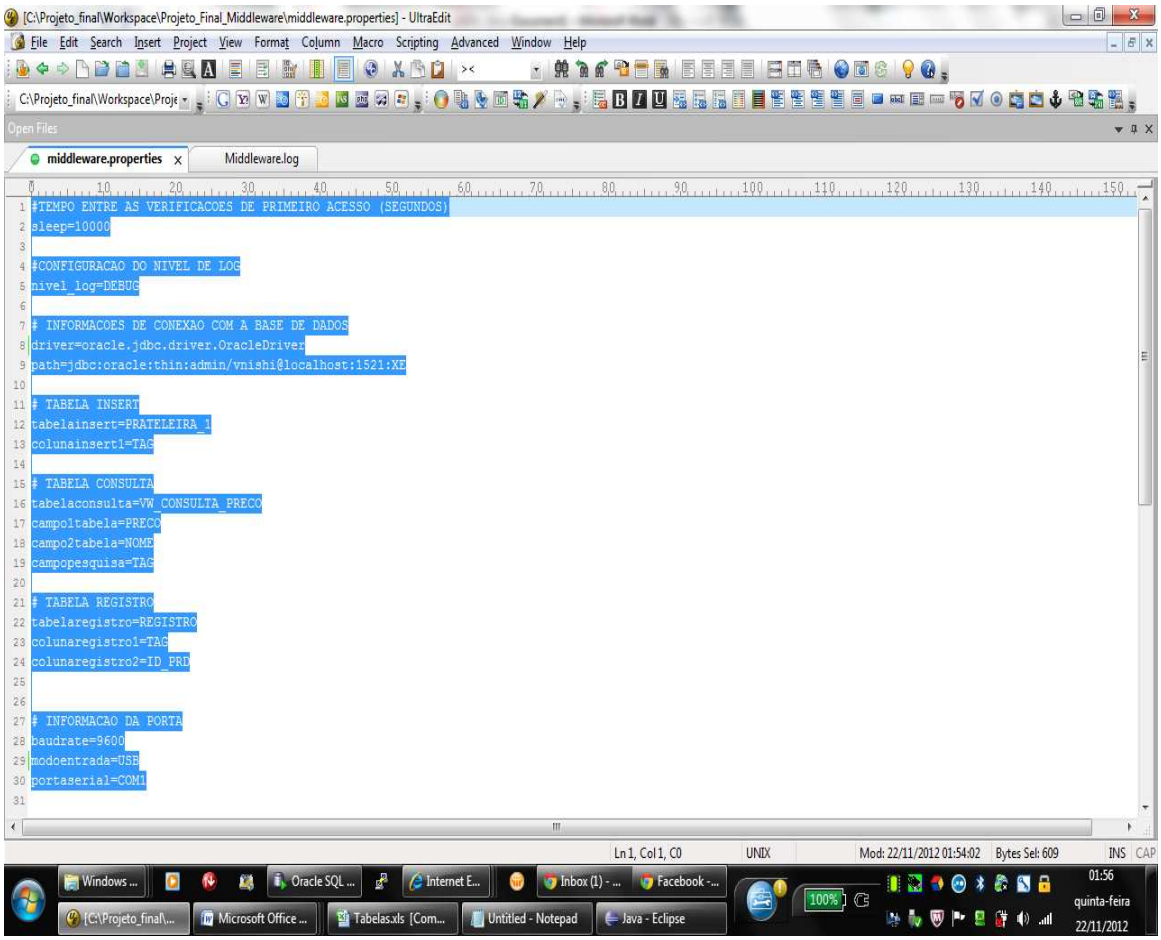

Figura C. 1– Arquivo parametrizado
Fonte: Autor

Passo 2 - Validar que dado não existe no banco

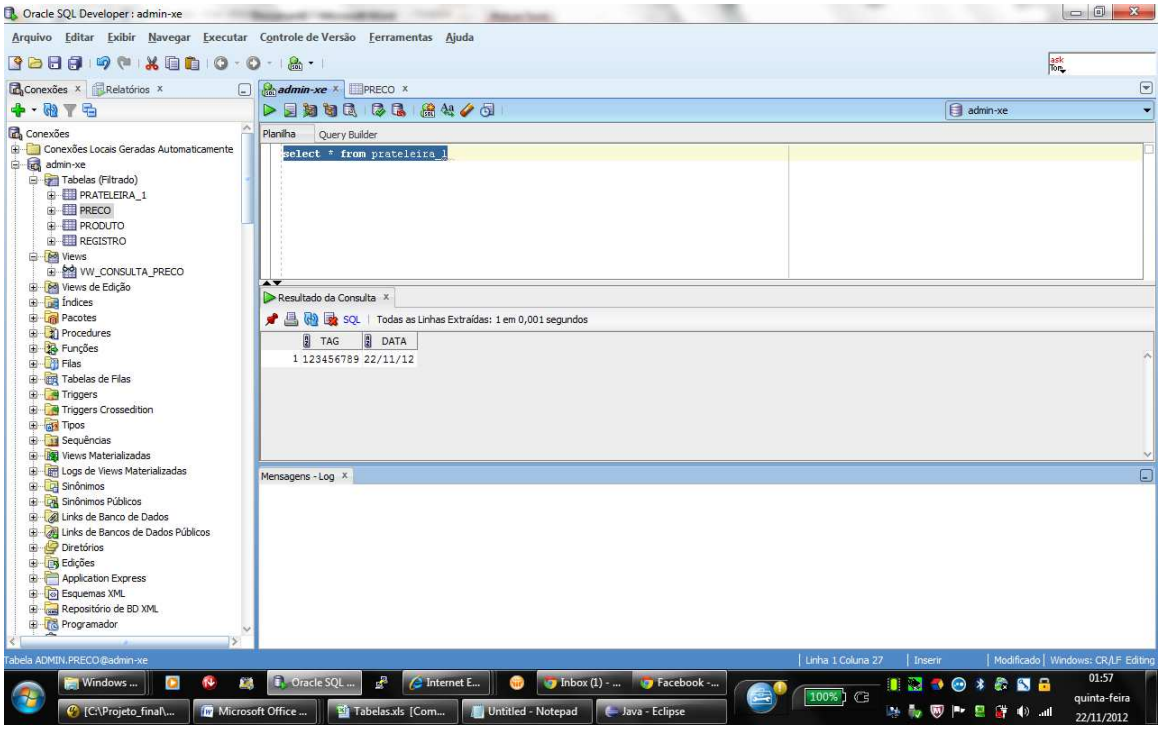

Figura C. 2– Validação
Fonte: Autor

Passo 3 - Iniciar Middleware;

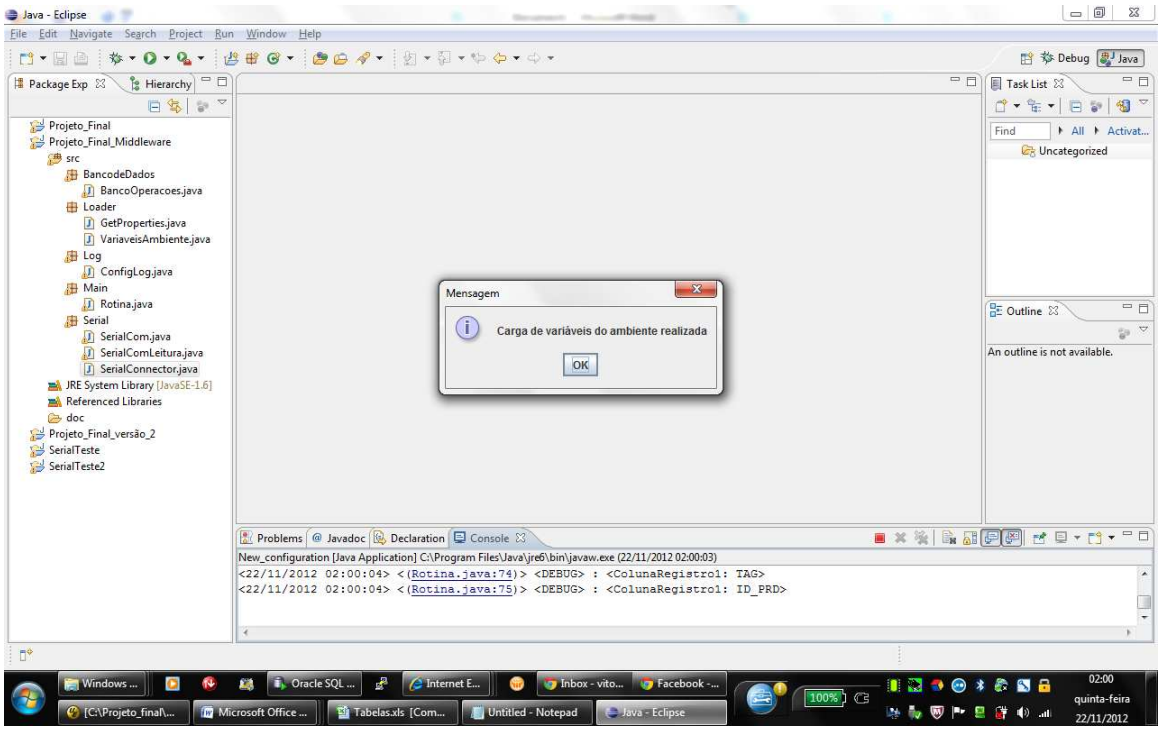

Figura C. 3– Inicio Carga
Fonte: Autor

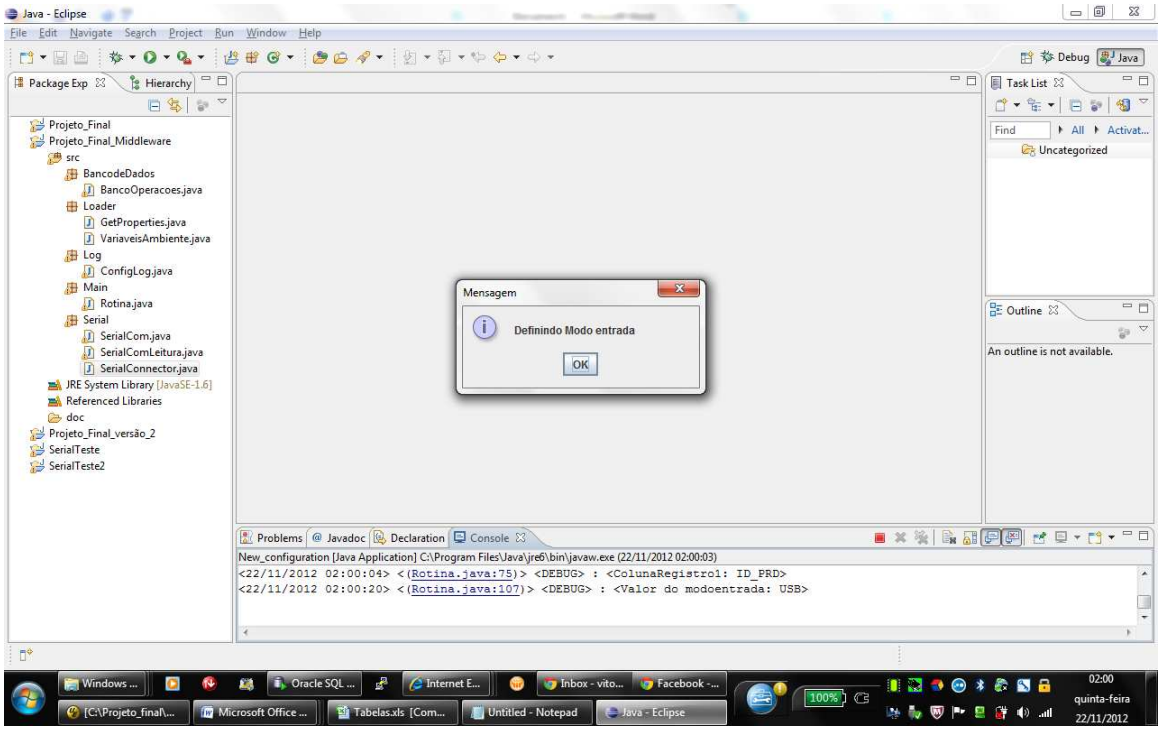

Figura C. 4– Modo Entrada
Fonte: Autor

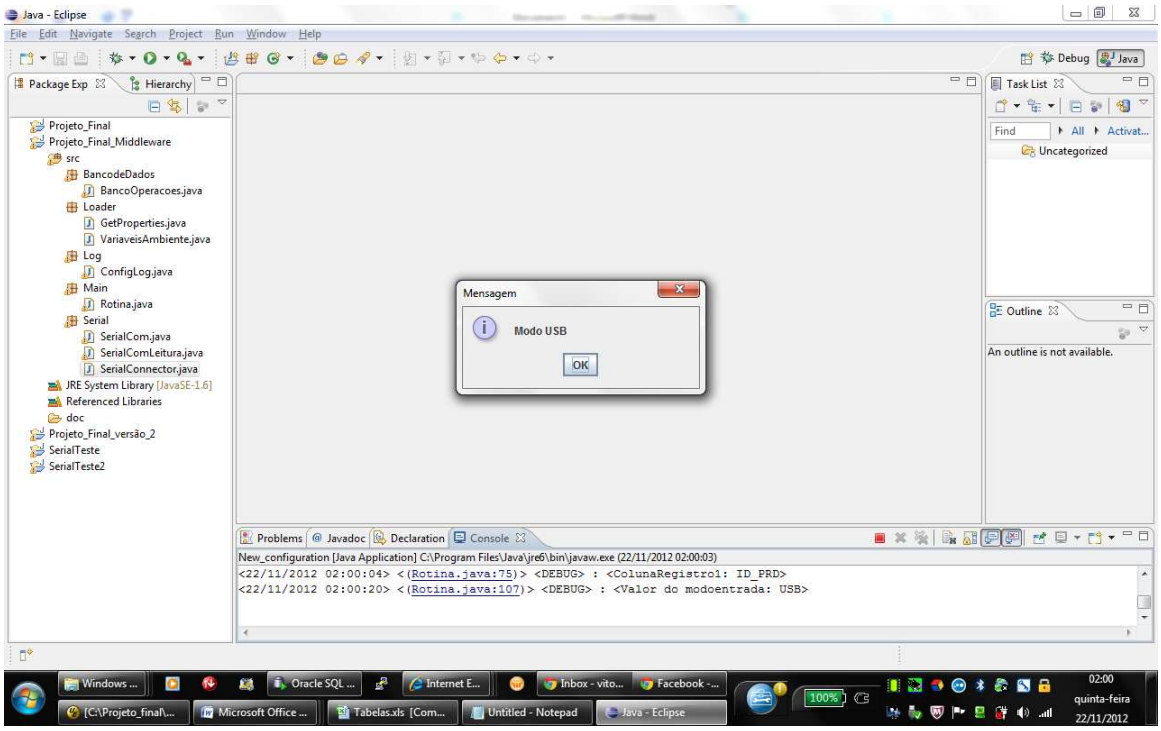

Figura C. 5– Modo USB
Fonte: Autor

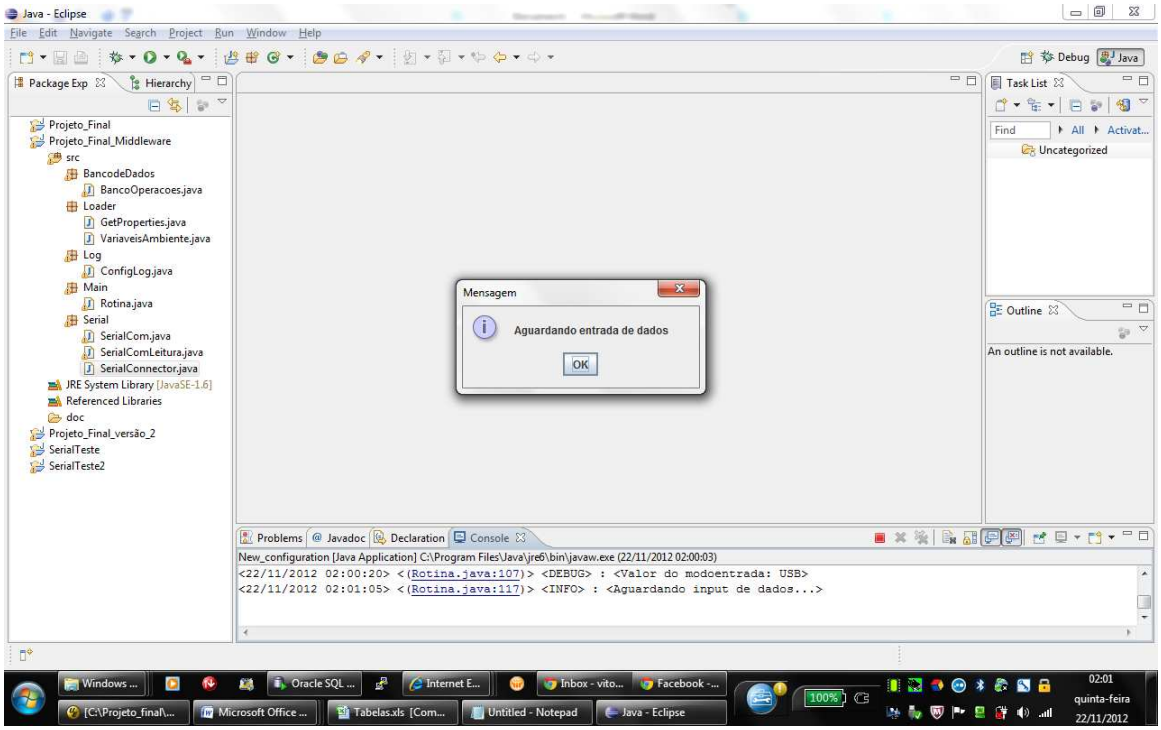

Figura C. 6– Aguardando dados
Fonte: Autor

Passo 4 - Enviar dado;

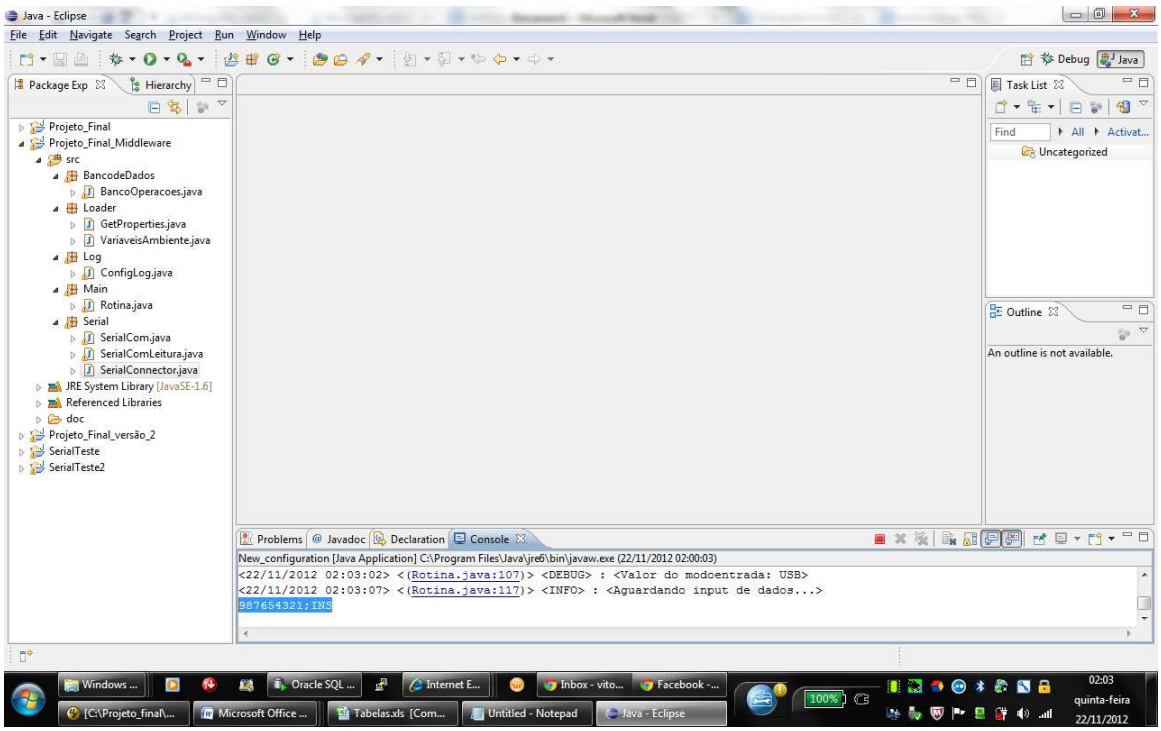

Figura C. 7– Enviar Dados
Fonte: Autor

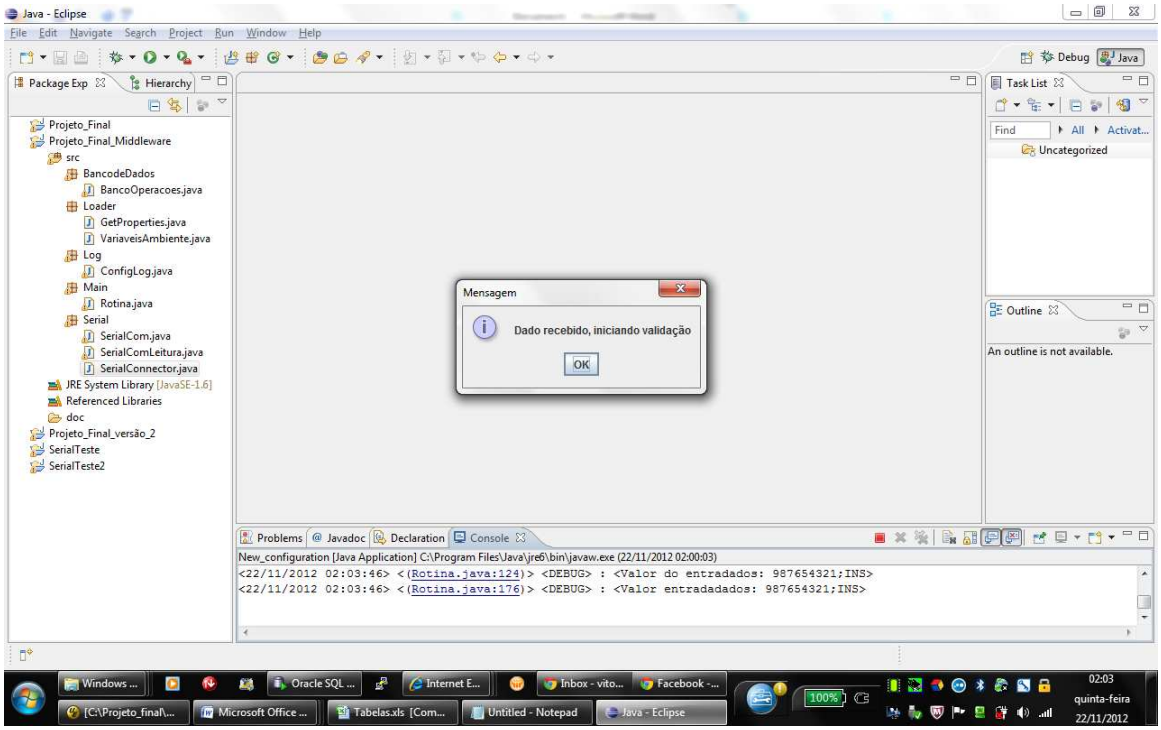

Figura C. 8– Dados Recebidos
Fonte: Autor

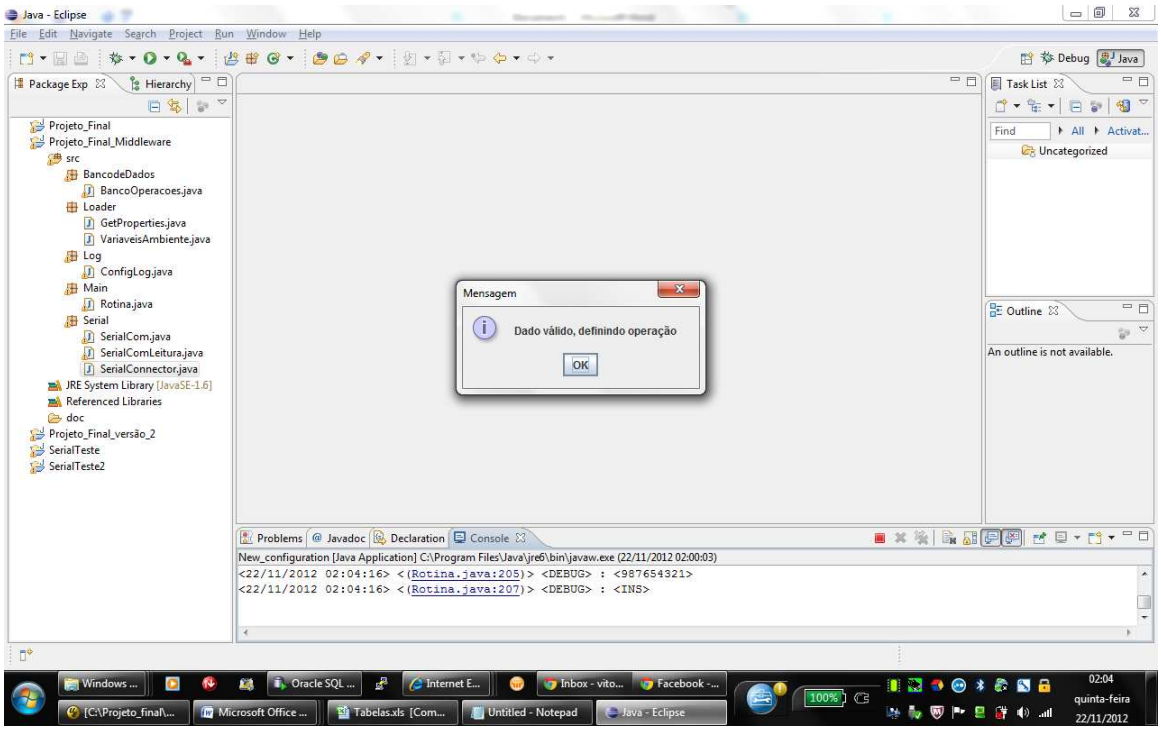

Figura C.9– Dado Válido
Fonte: Autor

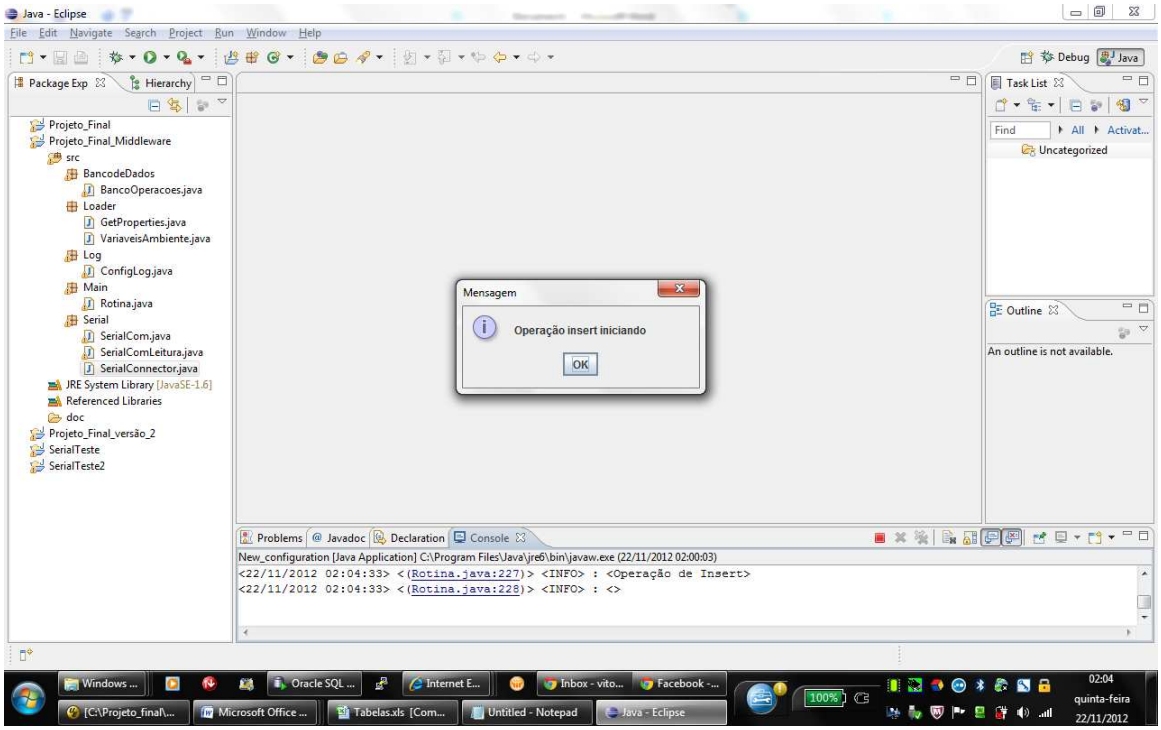

Figura C.10– Operação Insert
Fonte: Autor

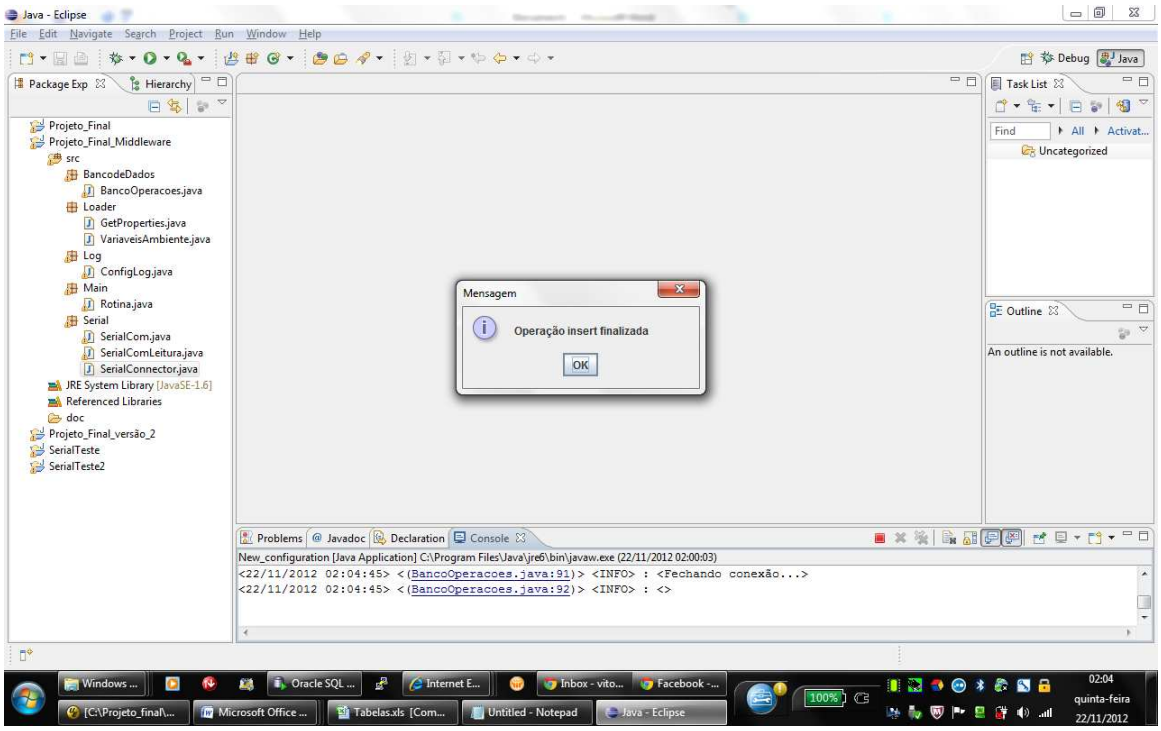

Figura C.11– Final Operação
Fonte: Autor

Passo 5 - Pesquisar dados no banco.

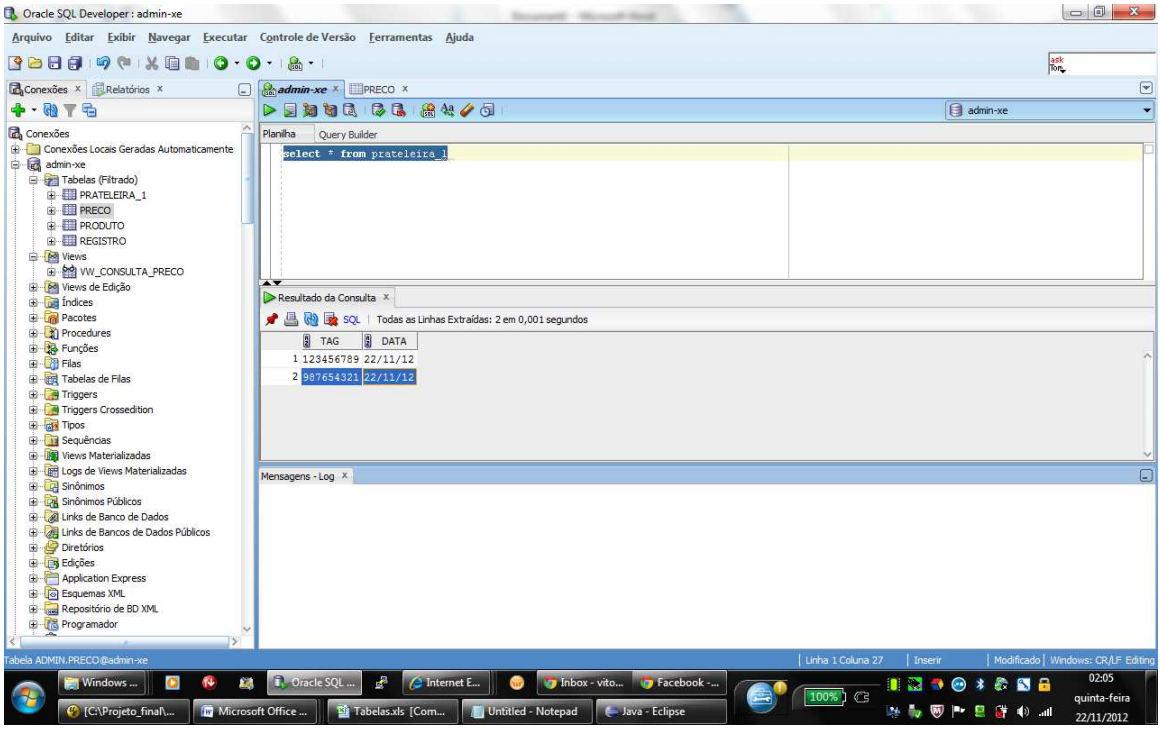

Figura C.12– Pesquisar
Fonte: Autor

Log complento do cenário:

<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: DEBUG>
<22/11/2012 01:59:14> <(Rotina.java:36)> <INFO> : <Nível de LOG setado para: DEBUG>
<22/11/2012 01:59:14> <(Rotina.java:37)> <INFO> : <>
<22/11/2012 01:59:14> <(Rotina.java:39)> <INFO> : <>
<22/11/2012 01:59:14> <(Rotina.java:40)> <INFO> : <Setando variáveis iniciais do módulo...>
<22/11/2012 01:59:14> <(Rotina.java:41)> <INFO> : <>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: oracle.jdbc.driver.OracleDriver>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: VW_CONSULTA_PRECO>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRATELEIRA_1>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: REGISTRO>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRECO>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: NOME>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 10000>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 9600>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: COM1>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: USB>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>

<22/11/2012 01:59:14> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: ID_PRD>
<22/11/2012 01:59:14> <(Rotina.java:59)> <DEBUG> : <Driver: oracle.jdbc.driver.OracleDriver>
<22/11/2012 01:59:14> <(Rotina.java:60)> <DEBUG> : <Path: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 01:59:14> <(Rotina.java:61)> <DEBUG> : <BaudRate: 9600>
<22/11/2012 01:59:14> <(Rotina.java:62)> <DEBUG> : <SerialPorta: COM1>
<22/11/2012 01:59:14> <(Rotina.java:63)> <DEBUG> : <ModoEntrada: USB>
<22/11/2012 01:59:14> <(Rotina.java:64)> <DEBUG> : <SleepTime: 10000>
<22/11/2012 01:59:14> <(Rotina.java:65)> <DEBUG> : <Tabela Consulta: VW_CONSULTA_PRECO>
<22/11/2012 01:59:14> <(Rotina.java:66)> <DEBUG> : <Tabela Insert: PRATELEIRA_1>
<22/11/2012 01:59:14> <(Rotina.java:67)> <DEBUG> : <Tabela Registro: REGISTRO>
<22/11/2012 01:59:14> <(Rotina.java:68)> <DEBUG> : <Campo da tabela: PRECO>
<22/11/2012 01:59:14> <(Rotina.java:69)> <DEBUG> : <Campo da tabela: NOME>
<22/11/2012 01:59:14> <(Rotina.java:70)> <DEBUG> : <Dados1tabela: null>
<22/11/2012 01:59:14> <(Rotina.java:71)> <DEBUG> : <Dados2tabela: null>
<22/11/2012 01:59:14> <(Rotina.java:72)> <DEBUG> : <CampoPesquisa: TAG>
<22/11/2012 01:59:14> <(Rotina.java:73)> <DEBUG> : <ColunaInsert1: TAG>
<22/11/2012 01:59:14> <(Rotina.java:74)> <DEBUG> : <ColunaRegistro1: TAG>
<22/11/2012 01:59:14> <(Rotina.java:75)> <DEBUG> : <ColunaRegistro1: ID_PRD>
<22/11/2012 01:59:16> <(Rotina.java:107)> <DEBUG> : <Valor do modoentrada: USB>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: DEBUG>
<22/11/2012 02:00:03> <(Rotina.java:36)> <INFO> : <Nível de LOG setado para: DEBUG>
<22/11/2012 02:00:03> <(Rotina.java:37)> <INFO> : <>
<22/11/2012 02:00:03> <(Rotina.java:39)> <INFO> : <>
<22/11/2012 02:00:03> <(Rotina.java:40)> <INFO> : <Setando variáveis iniciais do módulo...>
<22/11/2012 02:00:03> <(Rotina.java:41)> <INFO> : <>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: oracle.jdbc.driver.OracleDriver>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: VW_CONSULTA_PRECO>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRATELEIRA_1>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: REGISTRO>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRECO>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: NOME>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 10000>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 9600>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: COM1>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: USB>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 02:00:03> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: ID_PRD>
<22/11/2012 02:00:03> <(Rotina.java:59)> <DEBUG> : <Driver: oracle.jdbc.driver.OracleDriver>
<22/11/2012 02:00:03> <(Rotina.java:60)> <DEBUG> : <Path: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 02:00:03> <(Rotina.java:61)> <DEBUG> : <BaudRate: 9600>
<22/11/2012 02:00:03> <(Rotina.java:62)> <DEBUG> : <SerialPorta: COM1>
<22/11/2012 02:00:03> <(Rotina.java:63)> <DEBUG> : <ModoEntrada: USB>
<22/11/2012 02:00:03> <(Rotina.java:64)> <DEBUG> : <SleepTime: 10000>
<22/11/2012 02:00:03> <(Rotina.java:65)> <DEBUG> : <Tabela Consulta: VW_CONSULTA_PRECO>
<22/11/2012 02:00:03> <(Rotina.java:66)> <DEBUG> : <Tabela Insert: PRATELEIRA_1>
<22/11/2012 02:00:03> <(Rotina.java:67)> <DEBUG> : <Tabela Registro: REGISTRO>

<22/11/2012 02:00:03> <(Rotina.java:68)> <DEBUG> : <Campo da tabela: PRECO>
<22/11/2012 02:00:03> <(Rotina.java:69)> <DEBUG> : <Campo da tabela: NOME>
<22/11/2012 02:00:04> <(Rotina.java:70)> <DEBUG> : <Dados1tabela: null>
<22/11/2012 02:00:04> <(Rotina.java:71)> <DEBUG> : <Dados2tabela: null>
<22/11/2012 02:00:04> <(Rotina.java:72)> <DEBUG> : <CampoPesquisa: TAG>
<22/11/2012 02:00:04> <(Rotina.java:73)> <DEBUG> : <ColunaInsert1: TAG>
<22/11/2012 02:00:04> <(Rotina.java:74)> <DEBUG> : <ColunaRegistro1: TAG>
<22/11/2012 02:00:04> <(Rotina.java:75)> <DEBUG> : <ColunaRegistro1: ID_PRD>
<22/11/2012 02:00:20> <(Rotina.java:107)> <DEBUG> : <Valor do modoentrada: USB>
<22/11/2012 02:01:05> <(Rotina.java:117)> <INFO> : <Aguardando input de dados...>
<22/11/2012 02:02:19> <(SerialConnector.java:188)> <DEBUG> : <Valor do str: 987654321>
<22/11/2012 02:02:19> <(Rotina.java:124)> <DEBUG> : <Valor do entradados: 987654321>
<22/11/2012 02:02:19> <(Rotina.java:176)> <DEBUG> : <Valor entradadados: 987654321>
<22/11/2012 02:03:02> <(Rotina.java:186)> <DEBUG> : <Testetoken é: 987654321>
<22/11/2012 02:03:02> <(Rotina.java:190)> <INFO> : <>
<22/11/2012 02:03:02> <(Rotina.java:191)> <INFO> : <Input enviado esta errado!!!>
<22/11/2012 02:03:02> <(Rotina.java:192)> <INFO> : <Contém mais de 2 parametros separados por ;>
<22/11/2012 02:03:02> <(Rotina.java:193)> <INFO> : <Deve ser enviado nova string>
<22/11/2012 02:03:02> <(Rotina.java:194)> <INFO> : <>
<22/11/2012 02:03:02> <(Rotina.java:107)> <DEBUG> : <Valor do modoentrada: USB>
<22/11/2012 02:03:07> <(Rotina.java:117)> <INFO> : <Aguardando input de dados...>
<22/11/2012 02:03:46> <(SerialConnector.java:188)> <DEBUG> : <Valor do str: 987654321;INS>
<22/11/2012 02:03:46> <(Rotina.java:124)> <DEBUG> : <Valor do entradados: 987654321;INS>
<22/11/2012 02:03:46> <(Rotina.java:176)> <DEBUG> : <Valor entradadados: 987654321;INS>
<22/11/2012 02:04:16> <(Rotina.java:186)> <DEBUG> : <Testetoken é: 987654321;INS>
<22/11/2012 02:04:16> <(Rotina.java:205)> <DEBUG> : <987654321>
<22/11/2012 02:04:16> <(Rotina.java:207)> <DEBUG> : <INS>
<22/11/2012 02:04:33> <(Rotina.java:226)> <INFO> : <>
<22/11/2012 02:04:33> <(Rotina.java:227)> <INFO> : <Operação de Insert>
<22/11/2012 02:04:33> <(Rotina.java:228)> <INFO> : <>
<22/11/2012 02:04:45> <(BancoOperacoes.java:56)> <INFO> : <>
<22/11/2012 02:04:45> <(BancoOperacoes.java:57)> <INFO> : <Setando variaveis para gravar no banco...>
<22/11/2012 02:04:45> <(BancoOperacoes.java:58)> <INFO> : <>
<22/11/2012 02:04:45> <(BancoOperacoes.java:59)> <INFO> : <Abrindo conexão...>
<22/11/2012 02:04:45> <(BancoOperacoes.java:60)> <INFO> : <>
<22/11/2012 02:04:45> <(BancoOperacoes.java:63)> <DEBUG> : <Driveroracle.jdbc.driver.OracleDriver>
<22/11/2012 02:04:45> <(BancoOperacoes.java:64)> <DEBUG> : <Pathjdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 02:04:45> <(BancoOperacoes.java:66)> <INFO> : <>
<22/11/2012 02:04:45> <(BancoOperacoes.java:67)> <INFO> : <populando dados do insert...>
<22/11/2012 02:04:45> <(BancoOperacoes.java:68)> <INFO> : <>
<22/11/2012 02:04:45> <(BancoOperacoes.java:71)> <DEBUG> : <SQL: INSERT INTO PRATELEIRA_1(TAG, DATA) VALUES (? , SYSDATE)>
<22/11/2012 02:04:45> <(BancoOperacoes.java:76)> <INFO> : <Executando insert...>
<22/11/2012 02:04:45> <(BancoOperacoes.java:77)> <INFO> : <>
<22/11/2012 02:04:45> <(BancoOperacoes.java:88)> <INFO> : <>
<22/11/2012 02:04:45> <(BancoOperacoes.java:89)> <INFO> : <Insert Executado com sucesso!>
<22/11/2012 02:04:45> <(BancoOperacoes.java:90)> <INFO> : <>
<22/11/2012 02:04:45> <(BancoOperacoes.java:91)> <INFO> : <Fechando conexão...>
<22/11/2012 02:04:45> <(BancoOperacoes.java:92)> <INFO> : <>

APÊNDICE D – Evidências Caso 3

Dados da Massa

Tag: 123498765
Dado adicional: 2

Dados Configuração BD

String conexão banco: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE
Tabela de Insert: REGISTRO
Coluna de Insert tag: TAG
Coluna de Insert dado adicional: ID_PRD

Dados Configuração Serial

Porta: COM1
Baudrate: 9600
Sleep: 10 segundos

Cenário: Enviar tag na porta serial com a operação REG

Passo 1 - Parametrizar arquivo middleware.properties

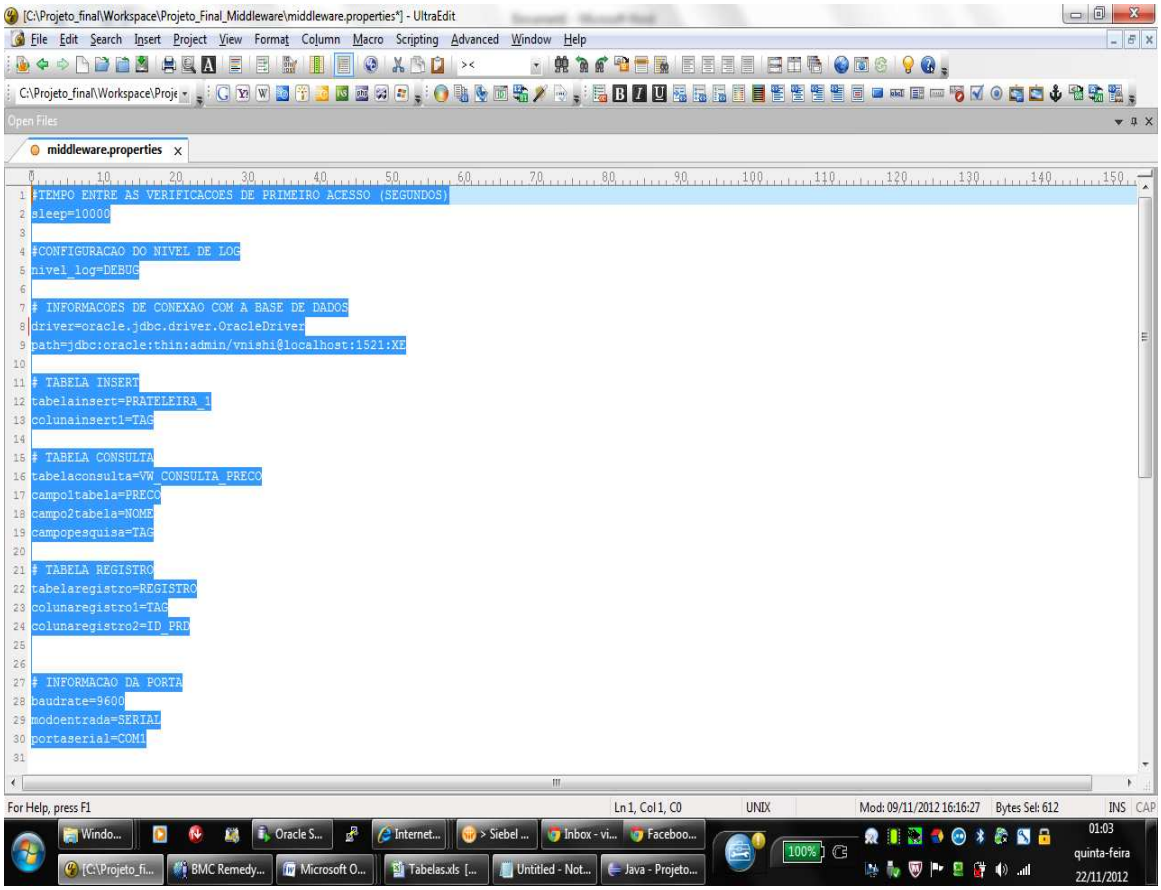

Figura D.1– Arquivo parametrizado
Fonte: Autor

Passo 2 - Validar que dado não existe no banco

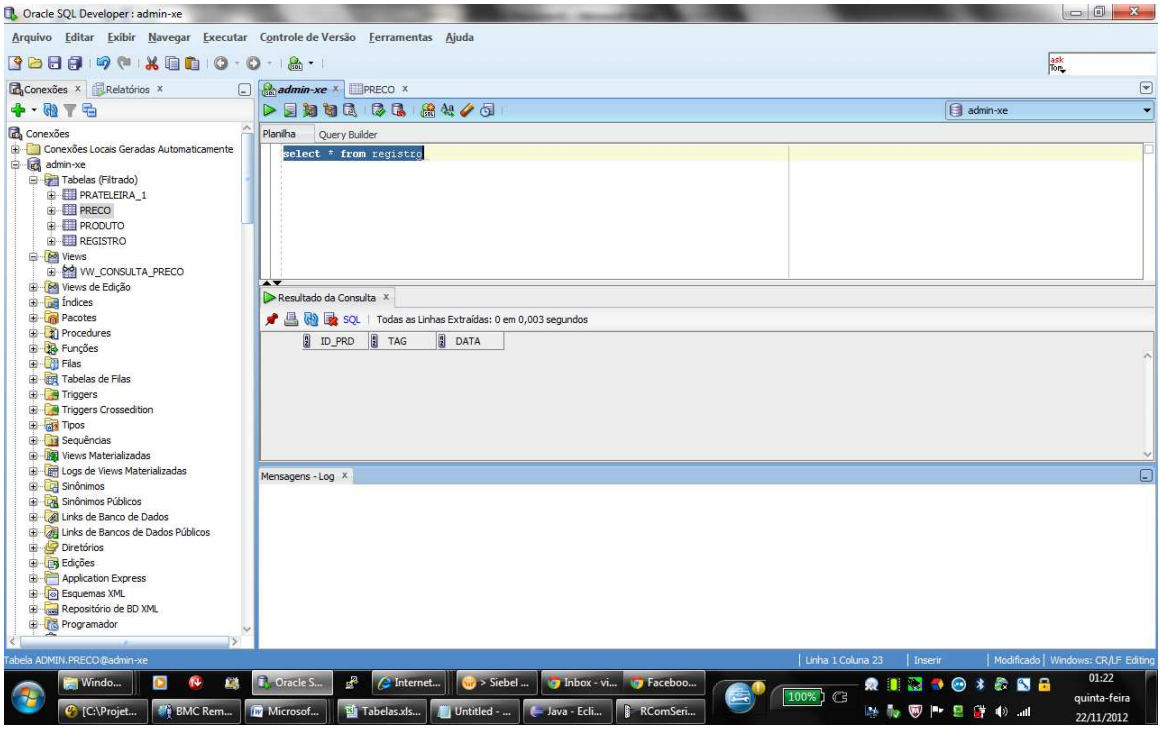

Figura D.2– Validação
Fonte: Autor

Passo 3 - Iniciar Middleware;

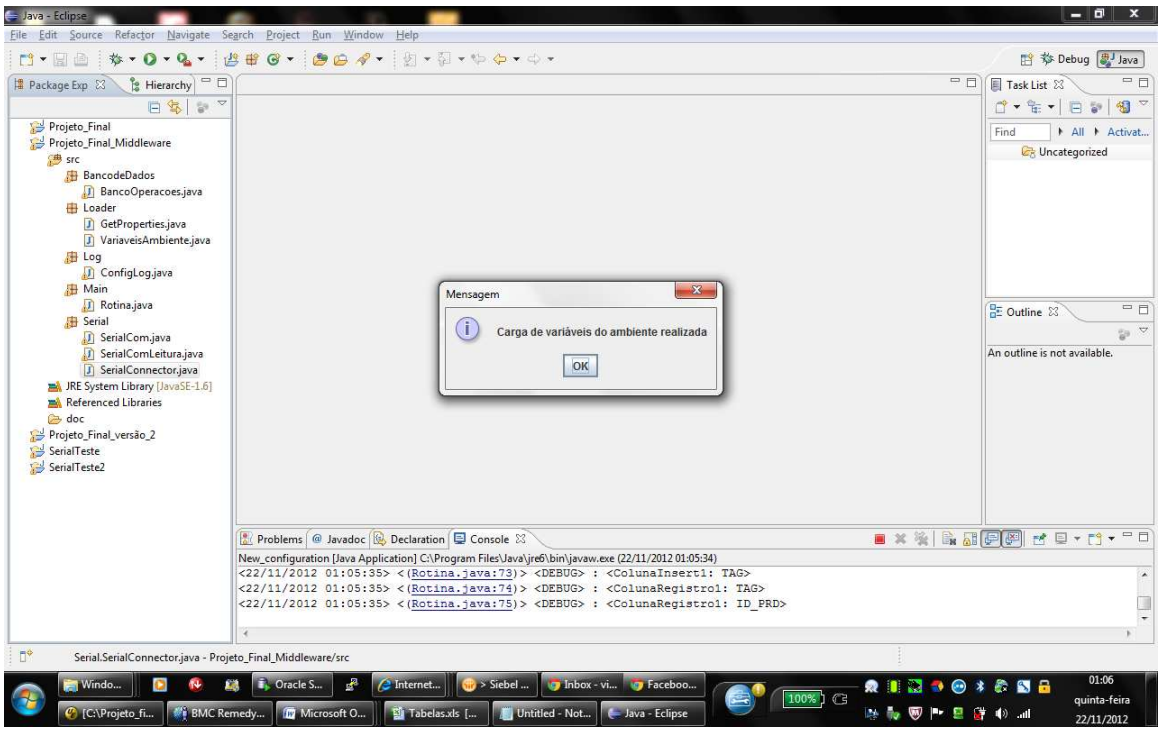

Figura D.3– Inicio Carga
Fonte: Autor

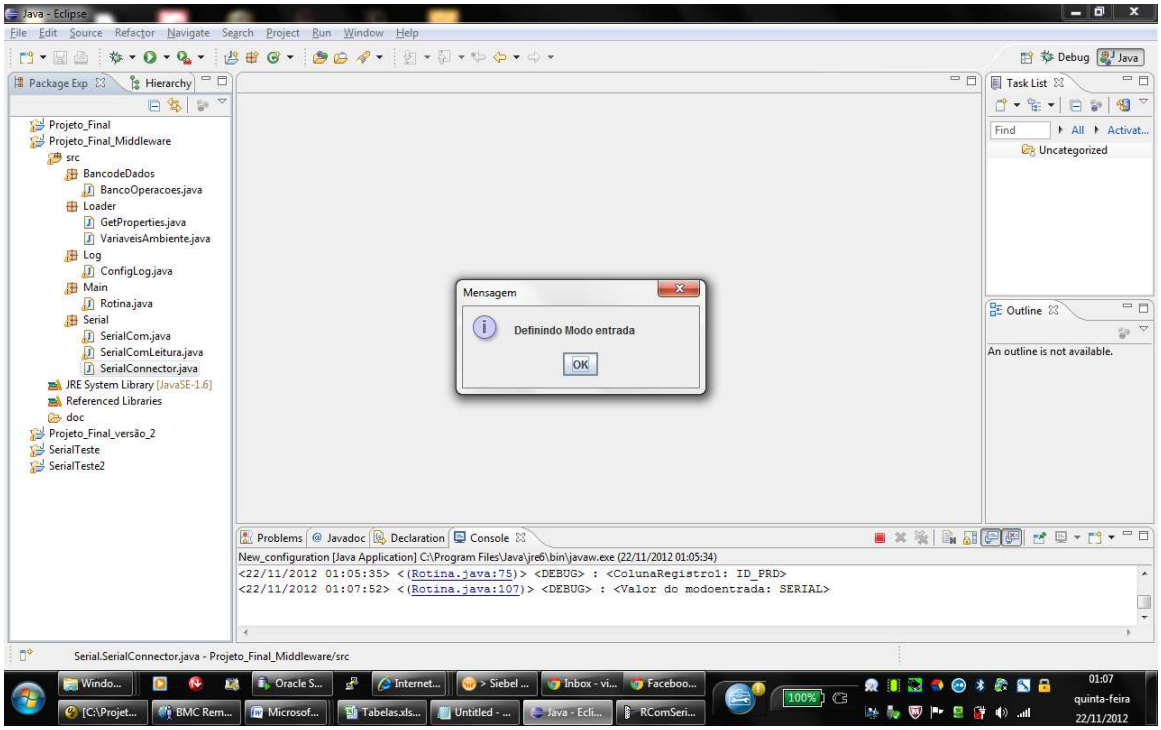

Figura D.4– Modo Entrada
Fonte: Autor

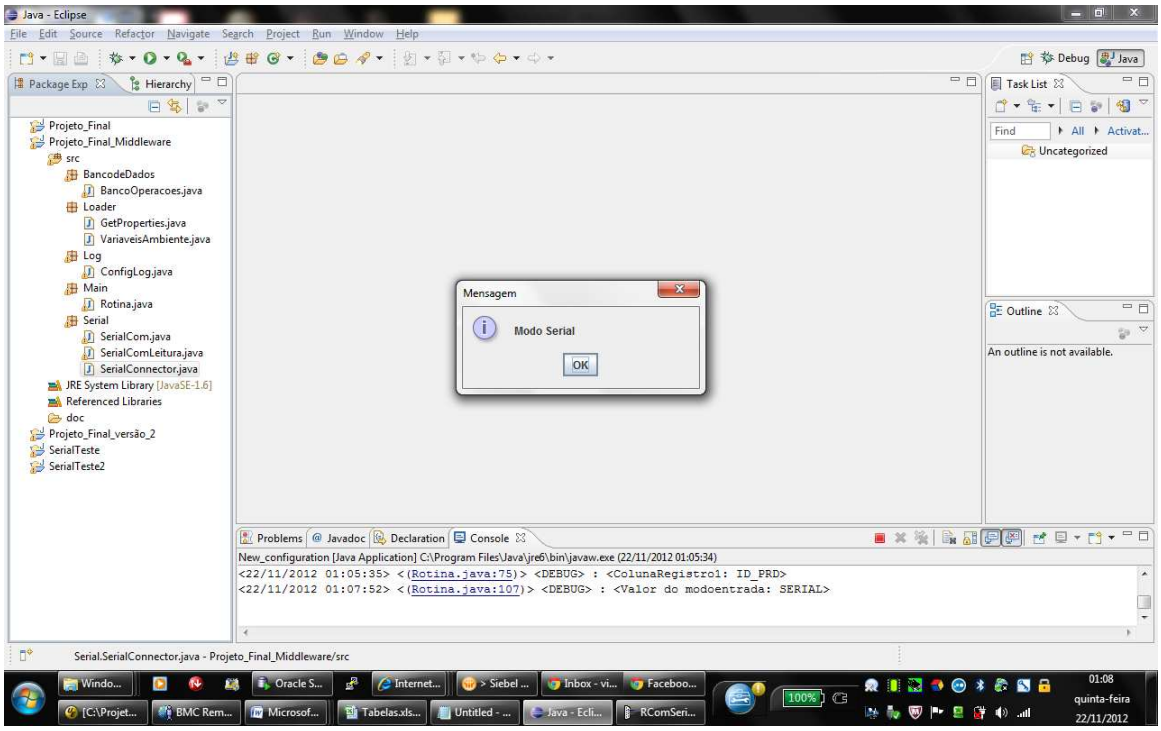

Figura D.5– Modo Serial
Fonte: Autor

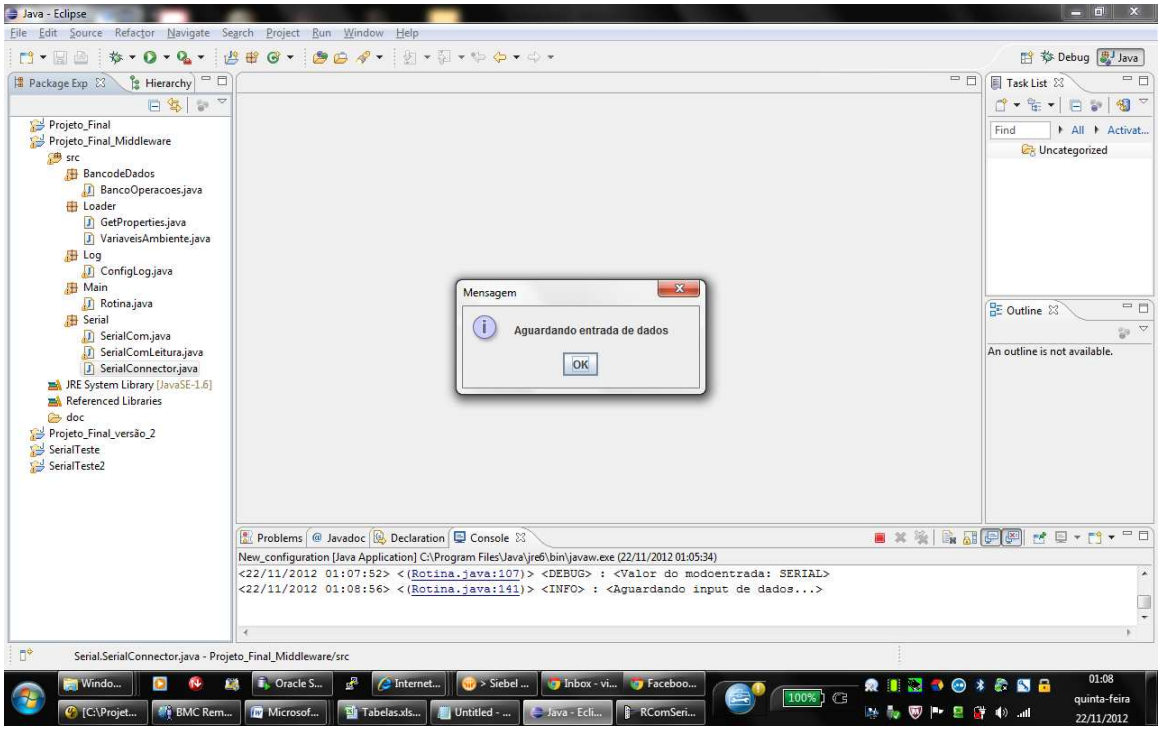

Figura D.6– Aguardando dados
Fonte: Autor

Passo 4 - Enviar dado;

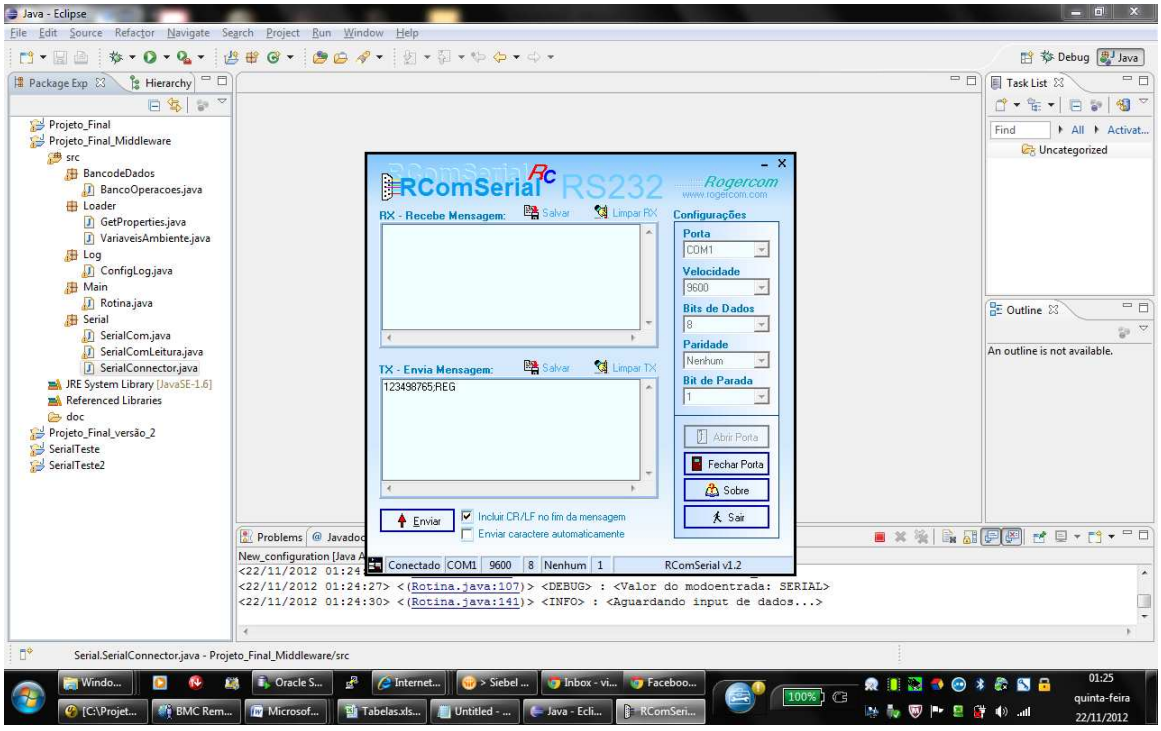

Figura D.7– Enviar Dados
Fonte: Autor

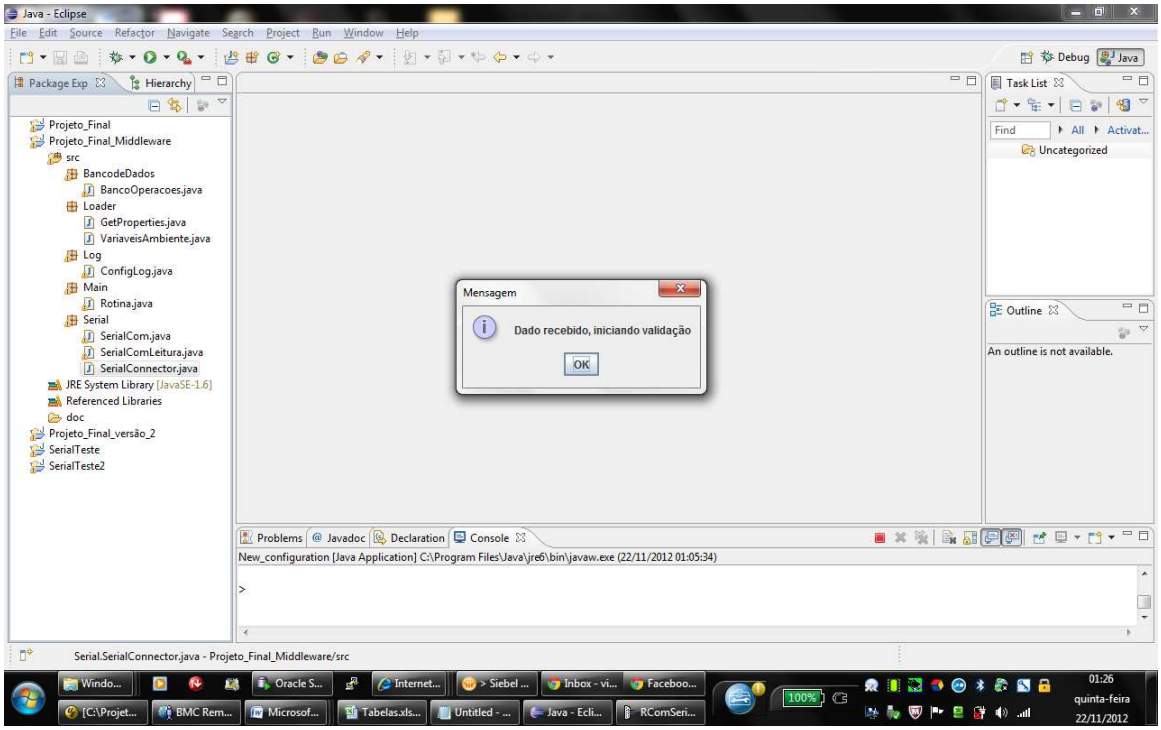

Figura D.8– Dados Recebidos
Fonte: Autor

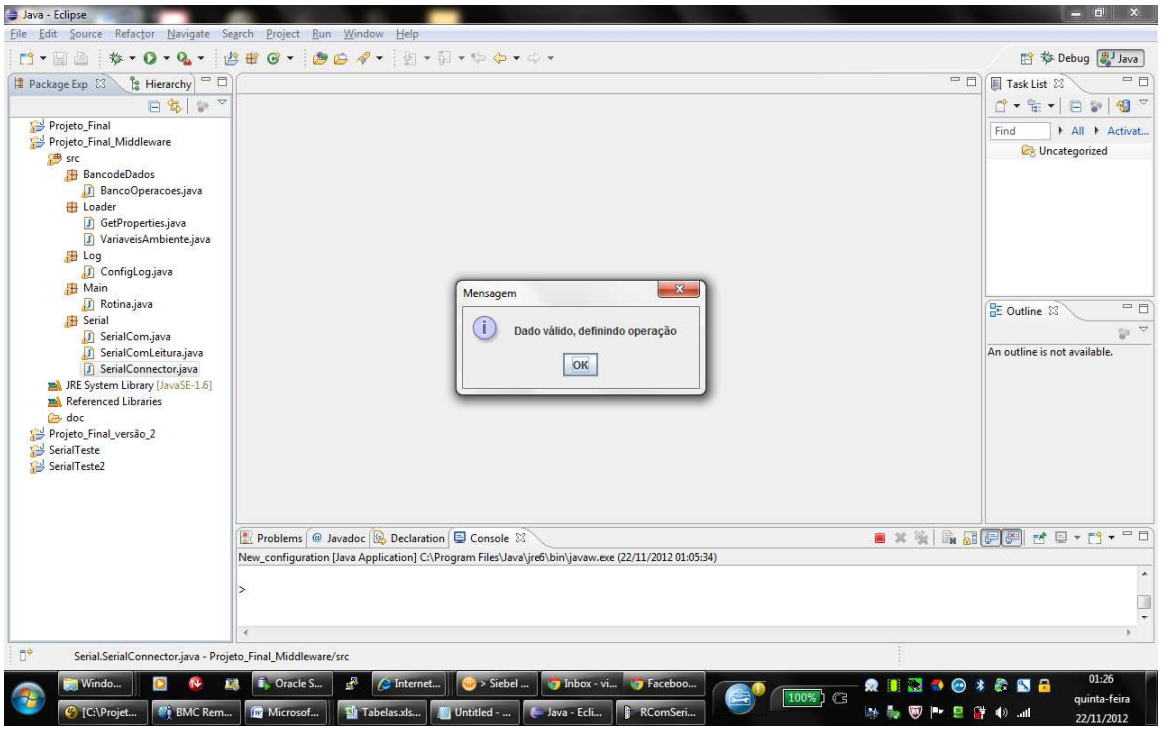

Figura D.9– Dado Válido
Fonte: Autor

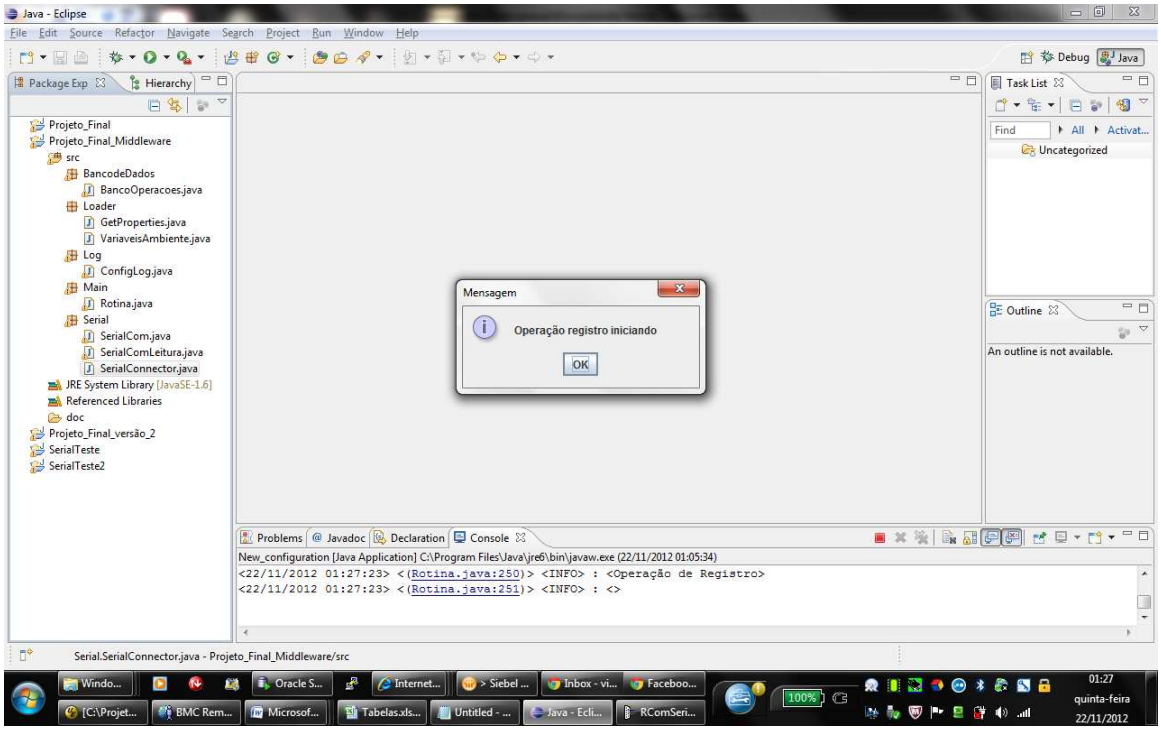

Figura D.10– Operação Registro
Fonte: Autor

Passo 5 - Acrescentar dado adicional;

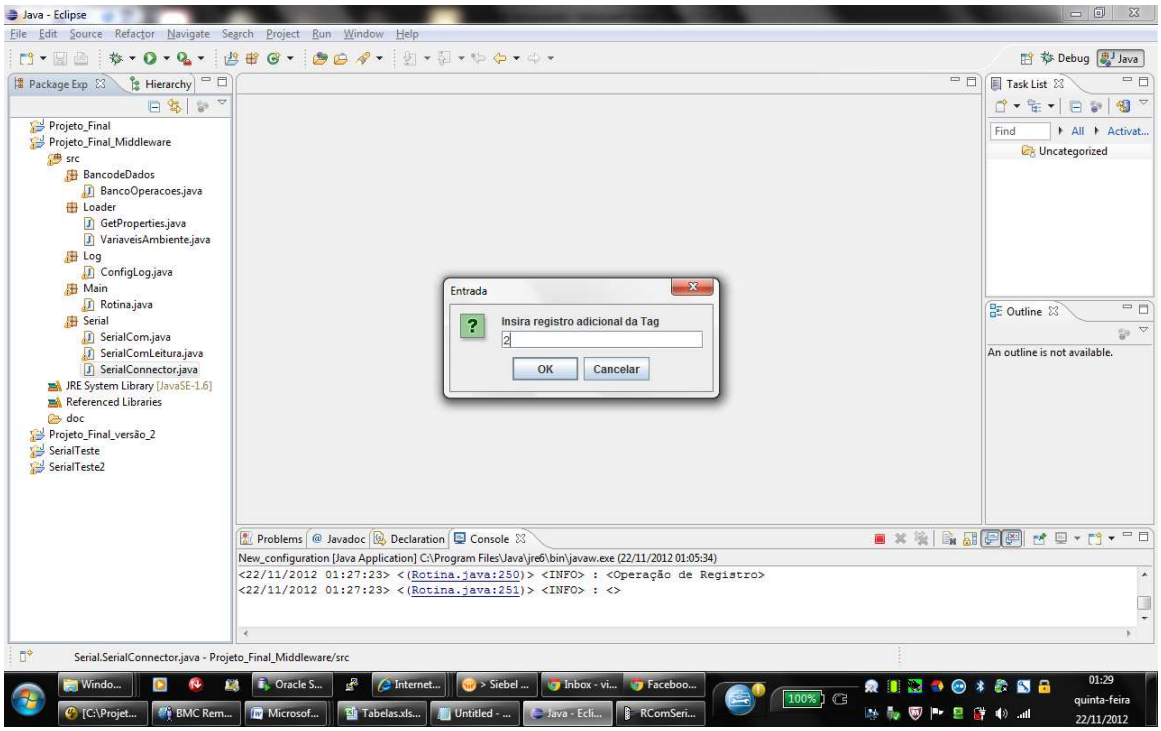

Figura D.11– Dado Adicional
Fonte: Autor

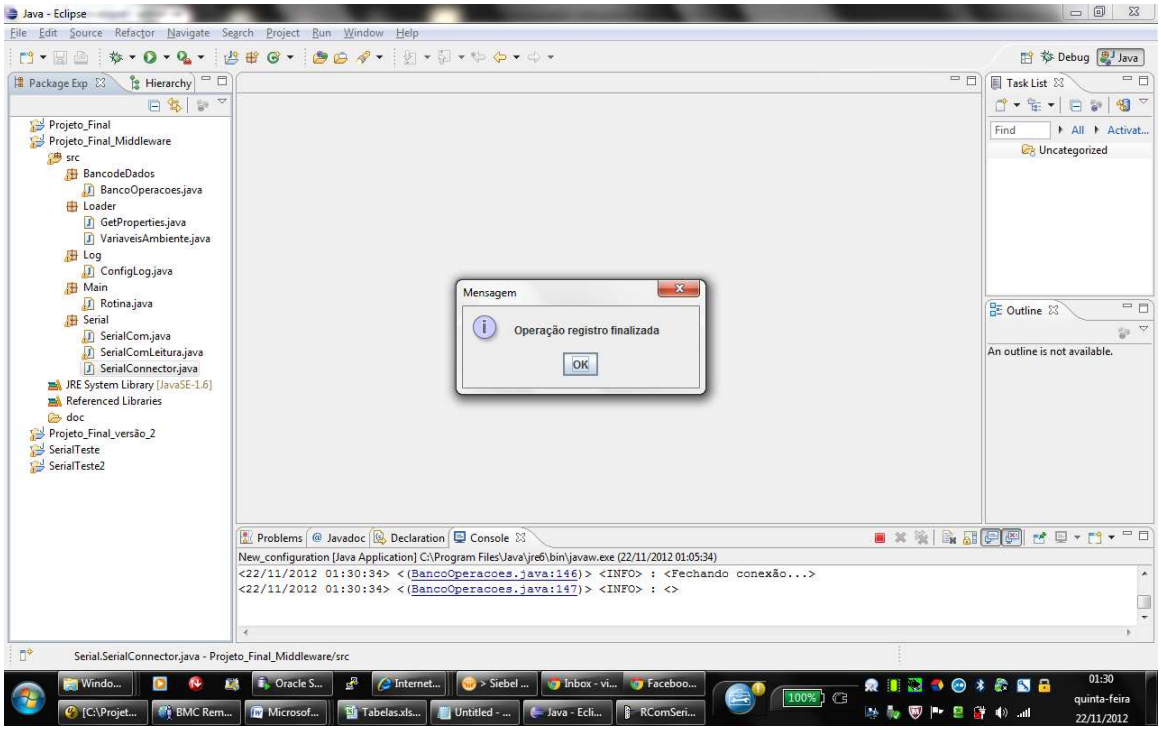

Figura D.12– Final Operação
Fonte: Autor

Passo 6 - Pesquisar dados no banco.

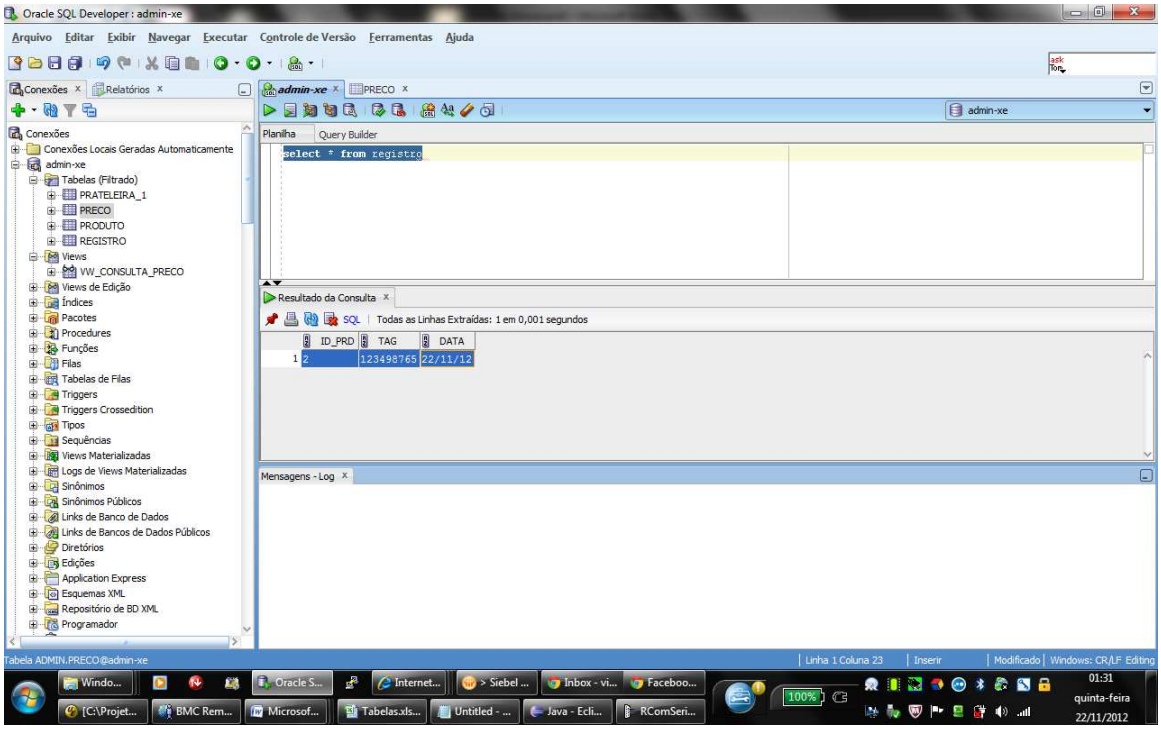

Figura D.13– Pesquisar
Fonte: Autor

Log completo do cenário:

<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: DEBUG>
<22/11/2012 01:24:25> <(Rotina.java:36)> <INFO> : <Nível de LOG setado para: DEBUG>
<22/11/2012 01:24:25> <(Rotina.java:37)> <INFO> : <>
<22/11/2012 01:24:25> <(Rotina.java:39)> <INFO> : <>
<22/11/2012 01:24:25> <(Rotina.java:40)> <INFO> : <Setando variáveis iniciais do módulo...>
<22/11/2012 01:24:25> <(Rotina.java:41)> <INFO> : <>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: oracle.jdbc.driver.OracleDriver>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: VW_CONSULTA_PRECO>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRATELEIRA_1>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: REGISTRO>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRECO>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: NOME>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 10000>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 9600>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: COM1>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: SERIAL>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:24:25> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: ID_PRD>
<22/11/2012 01:24:25> <(Rotina.java:59)> <DEBUG> : <Driver: oracle.jdbc.driver.OracleDriver>
<22/11/2012 01:24:25> <(Rotina.java:60)> <DEBUG> : <Path: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 01:24:25> <(Rotina.java:61)> <DEBUG> : <BaudRate: 9600>
<22/11/2012 01:24:25> <(Rotina.java:62)> <DEBUG> : <SerialPorta: COM1>
<22/11/2012 01:24:25> <(Rotina.java:63)> <DEBUG> : <ModoEntrada: SERIAL>
<22/11/2012 01:24:25> <(Rotina.java:64)> <DEBUG> : <SleepTime: 10000>
<22/11/2012 01:24:25> <(Rotina.java:65)> <DEBUG> : <Tabela Consulta: VW_CONSULTA_PRECO>
<22/11/2012 01:24:25> <(Rotina.java:66)> <DEBUG> : <Tabela Insert: PRATELEIRA_1>
<22/11/2012 01:24:25> <(Rotina.java:67)> <DEBUG> : <Tabela Registro: REGISTRO>
<22/11/2012 01:24:25> <(Rotina.java:68)> <DEBUG> : <Campo da tabela: PRECO>
<22/11/2012 01:24:25> <(Rotina.java:69)> <DEBUG> : <Campo da tabela: NOME>
<22/11/2012 01:24:25> <(Rotina.java:70)> <DEBUG> : <Dados1tabela: null>
<22/11/2012 01:24:25> <(Rotina.java:71)> <DEBUG> : <Dados2tabela: null>
<22/11/2012 01:24:25> <(Rotina.java:72)> <DEBUG> : <CampoPesquisa: TAG>
<22/11/2012 01:24:25> <(Rotina.java:73)> <DEBUG> : <ColunaInsert1: TAG>
<22/11/2012 01:24:25> <(Rotina.java:74)> <DEBUG> : <ColunaRegistro1: TAG>
<22/11/2012 01:24:25> <(Rotina.java:75)> <DEBUG> : <ColunaRegistro1: ID_PRD>
<22/11/2012 01:24:27> <(Rotina.java:107)> <DEBUG> : <Valor do modoentrada: SERIAL>
<22/11/2012 01:24:30> <(Rotina.java:141)> <INFO> : <Aguardando input de dados...>
<22/11/2012 01:25:17> <(SerialConnector.java:25)> <INFO> : <Processo Leitura inicio>
<22/11/2012 01:25:17> <(SerialConnector.java:28)> <INFO> : <Leitura habilitada>
<22/11/2012 01:25:17> <(SerialConnector.java:33)> <INFO> : <Id porta obtido>
<22/11/2012 01:25:17> <(SerialConnector.java:38)> <INFO> : <Porta aberta>
<22/11/2012 01:25:17> <(SerialComLeitura.java:155)> <DEBUG> : <Valor entrada null>
<22/11/2012 01:25:17> <(SerialComLeitura.java:175)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@1359c1b>

<22/11/2012 01:25:17> <(SerialComLeitura.java:196)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@1359c1b>
<22/11/2012 01:25:17> <(SerialConnector.java:43)> <INFO> : <Poolling de leitura iniciado>
<22/11/2012 01:25:19> <(SerialComLeitura.java:286)> <DEBUG> : <Valor do bufferLeitura: >
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 49>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 49>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 50>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 50>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 51>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 51>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 52>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 52>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 57>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 57>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 56>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 56>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 55>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 55>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 54>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 54>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 53>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 53>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 59>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 59>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 82>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 82>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 69>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 69>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 71>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 71>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 13>
<22/11/2012 01:25:19> <(SerialComLeitura.java:331)> <DEBUG> : <Valor do novoDado: 13>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 10>
<22/11/2012 01:25:19> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 10>
<22/11/2012 01:25:19> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: -1>
<22/11/2012 01:25:19> <(SerialComLeitura.java:350)> <DEBUG> : <Valor do buffer: 123498765;REG

>
<22/11/2012 01:25:27> <(SerialConnector.java:57)> <INFO> : <Poolling de leitura finalizado>
<22/11/2012 01:25:27> <(SerialConnector.java:83)> <INFO> : <Leitura finalizada>
<22/11/2012 01:25:27> <(SerialConnector.java:93)> <DEBUG> : <Valor do buffer: 123498765;REG

>
<22/11/2012 01:25:27> <(SerialConnector.java:94)> <INFO> : <<>>
<22/11/2012 01:25:27> <(Rotina.java:151)> <DEBUG> : <Valor do entradados: 123498765;REG

>
<22/11/2012 01:25:27> <(Rotina.java:176)> <DEBUG> : <Valor entradadados: 123498765;REG

>
<22/11/2012 01:26:42> <(Rotina.java:186)> <DEBUG> : <Testetoken é: 123498765;REG

>
<22/11/2012 01:26:42> <(Rotina.java:205)> <DEBUG> : <123498765>
<22/11/2012 01:26:42> <(Rotina.java:207)> <DEBUG> : <REG

>
<22/11/2012 01:27:23> <(Rotina.java:249)> <INFO> : <>
<22/11/2012 01:27:23> <(Rotina.java:250)> <INFO> : <Operação de Registro>
<22/11/2012 01:27:23> <(Rotina.java:251)> <INFO> : <>

<22/11/2012 01:30:34> <(BancoOperacoes.java:112)> <INFO> : <>
<22/11/2012 01:30:34> <(BancoOperacoes.java:113)> <INFO> : <Setando variaveis para gravar no banco...>
<22/11/2012 01:30:34> <(BancoOperacoes.java:114)> <INFO> : <>
<22/11/2012 01:30:34> <(BancoOperacoes.java:115)> <INFO> : <Abrindo conexão...>
<22/11/2012 01:30:34> <(BancoOperacoes.java:116)> <INFO> : <>
<22/11/2012 01:30:34> <(BancoOperacoes.java:120)> <INFO> : <>
<22/11/2012 01:30:34> <(BancoOperacoes.java:121)> <INFO> : <populando dados do insert...>
<22/11/2012 01:30:34> <(BancoOperacoes.java:122)> <INFO> : <>
<22/11/2012 01:30:34> <(BancoOperacoes.java:125)> <DEBUG> : <SQL: INSERT INTO REGISTRO (TAG,ID_PRD, DATA) VALUES (? , ?, SYSDATE)>
<22/11/2012 01:30:34> <(BancoOperacoes.java:131)> <INFO> : <Executando insert...>
<22/11/2012 01:30:34> <(BancoOperacoes.java:132)> <INFO> : <>
<22/11/2012 01:30:34> <(BancoOperacoes.java:143)> <INFO> : <>
<22/11/2012 01:30:34> <(BancoOperacoes.java:144)> <INFO> : <Insert Executado com sucesso!>
<22/11/2012 01:30:34> <(BancoOperacoes.java:145)> <INFO> : <>
<22/11/2012 01:30:34> <(BancoOperacoes.java:146)> <INFO> : <Fechando conexão...>
<22/11/2012 01:30:34> <(BancoOperacoes.java:147)> <INFO> : <>

APÊNDICE E – Evidências Caso 4

Dados da Massa

Tag: 987651234
Dado adicional: 1

Dados Configuração BD

String conexão banco: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE
Tabela de Insert: REGISTRO
Coluna de Insert tag: TAG
Coluna de Insert dado adicional: ID_PRD

Dados Configuração Serial

Porta: COM1
Baudrate: 9600
Sleep: 10 segundos

Cenário: Enviar tag na porta USB com a operação REG

Passo 1 - Parametrizar arquivo middleware.properties

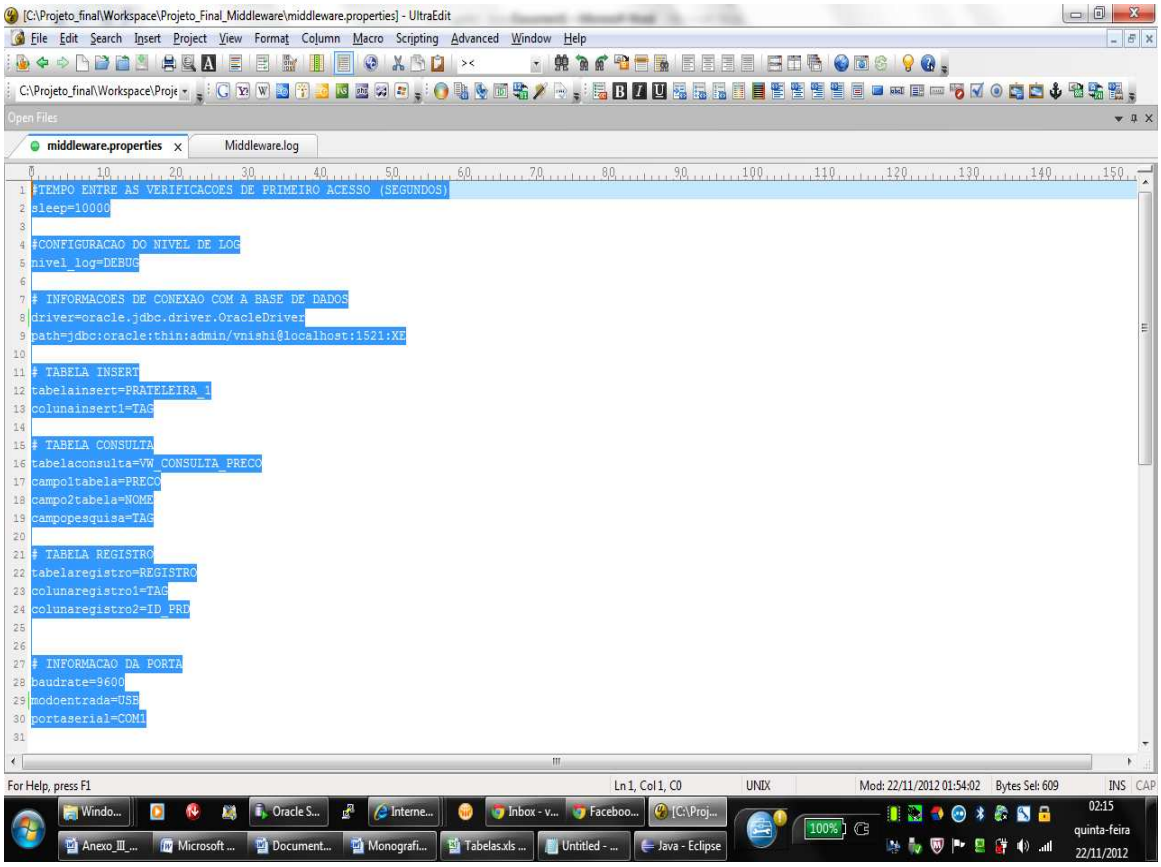

Figura E.1– Arquivo parametrizado
Fonte: Autor

Passo 2 - Validar que dado não existe no banco

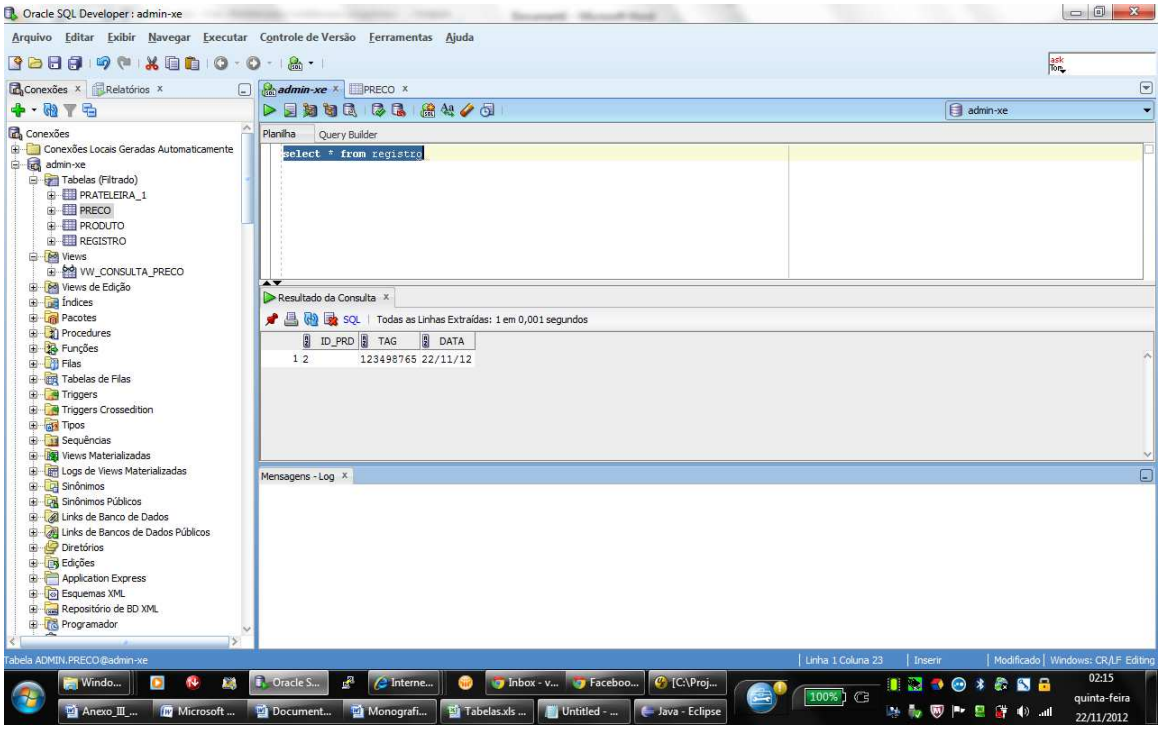

Figura E.2– Validação
Fonte: Autor

Passo 3 - Iniciar Middleware;

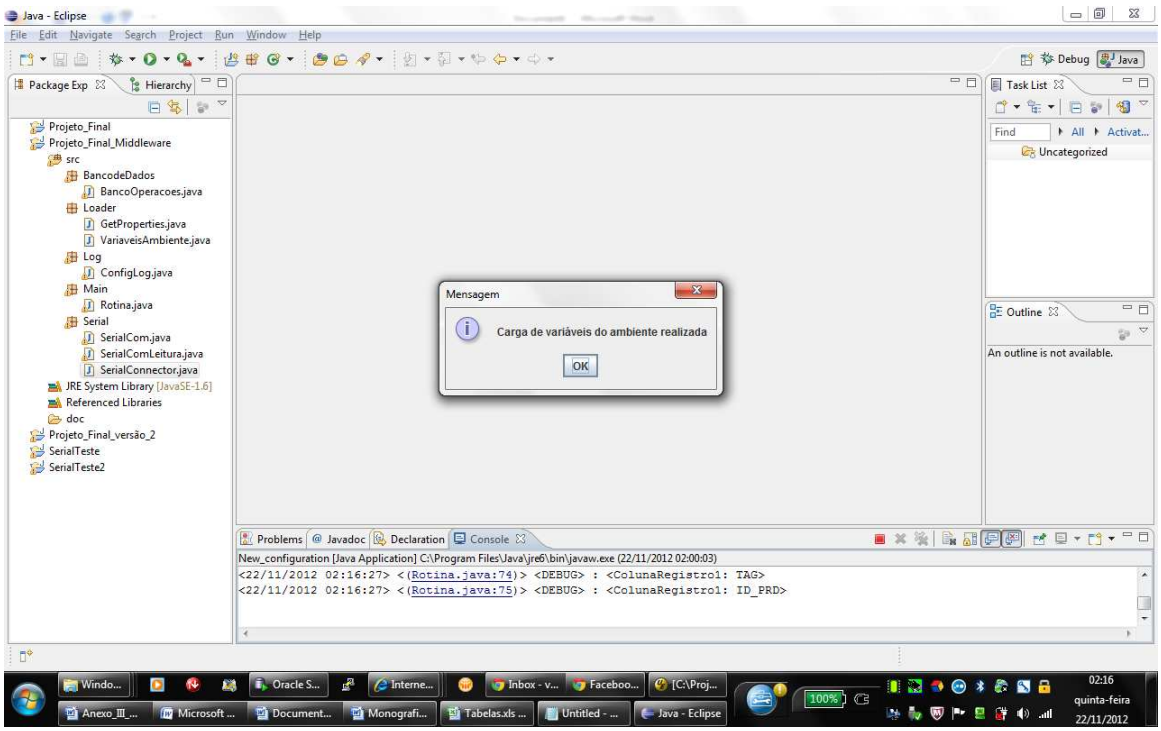

Figura E.3– Inicio Carga
Fonte: Autor

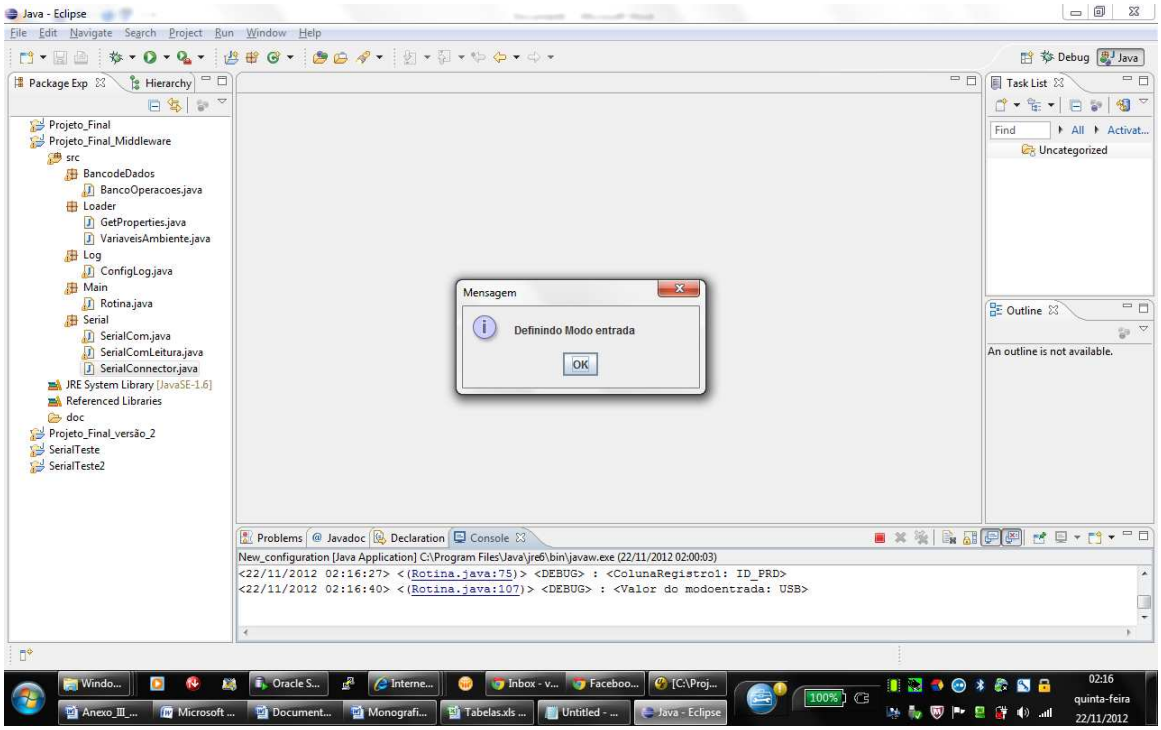

Figura E.4– Modo Entrada
Fonte: Autor

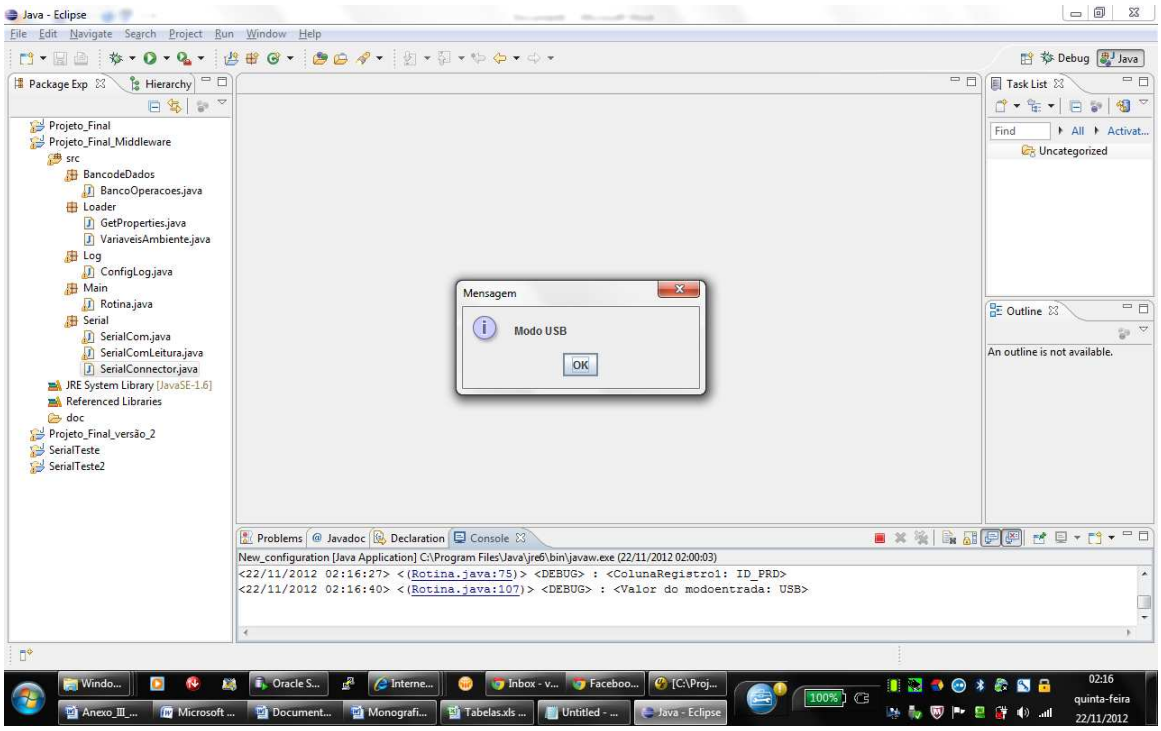

Figura E.5– Modo USB
Fonte: Autor

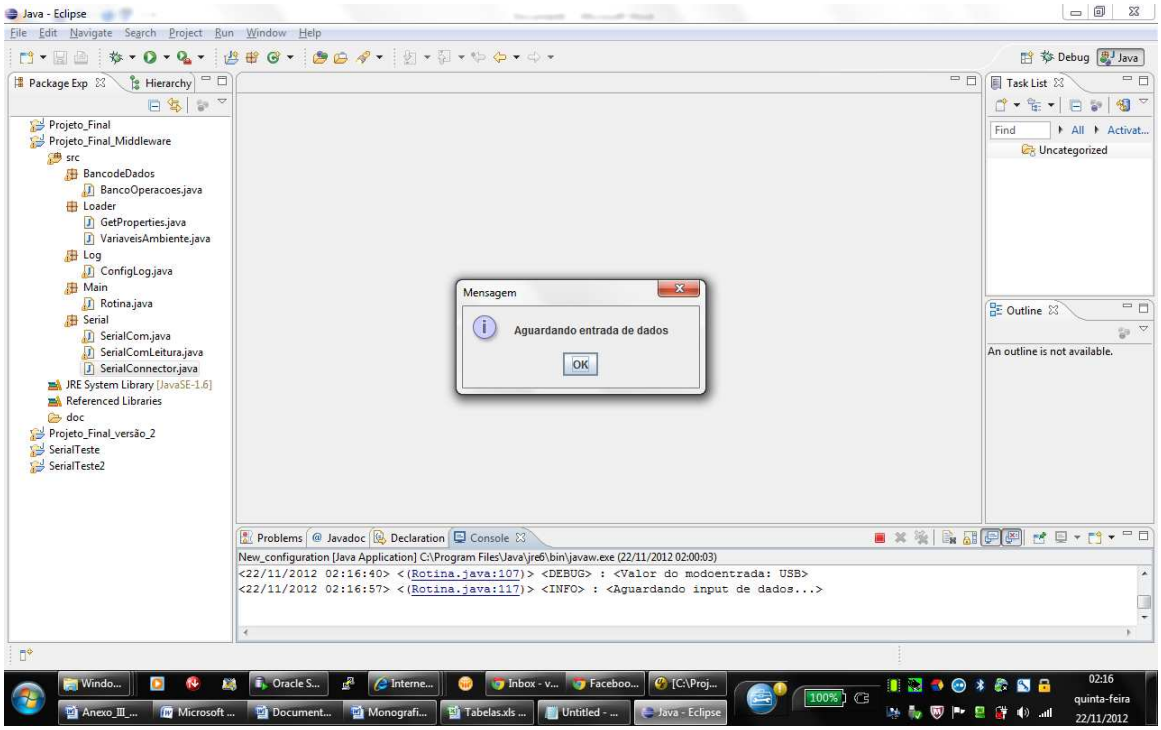

Figura E.6– Aguardando dados
Fonte: Autor

Passo 4 - Enviar dado;

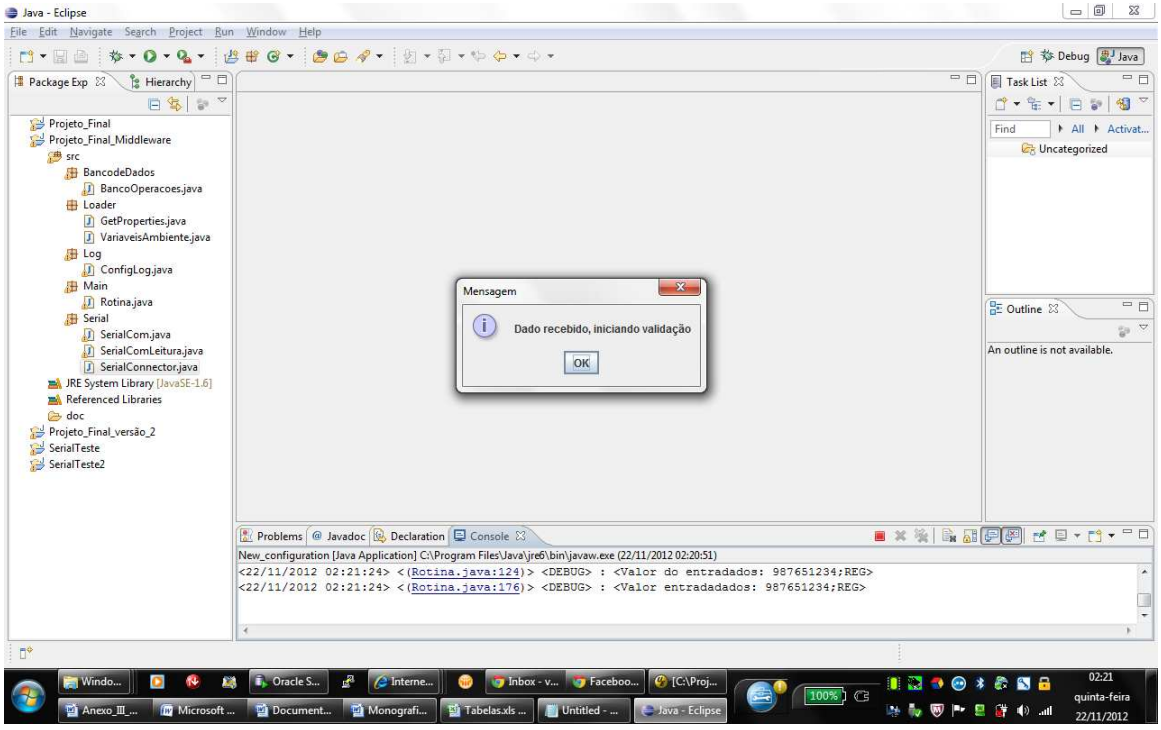

Figura E.7– Dados Recebidos
Fonte: Autor

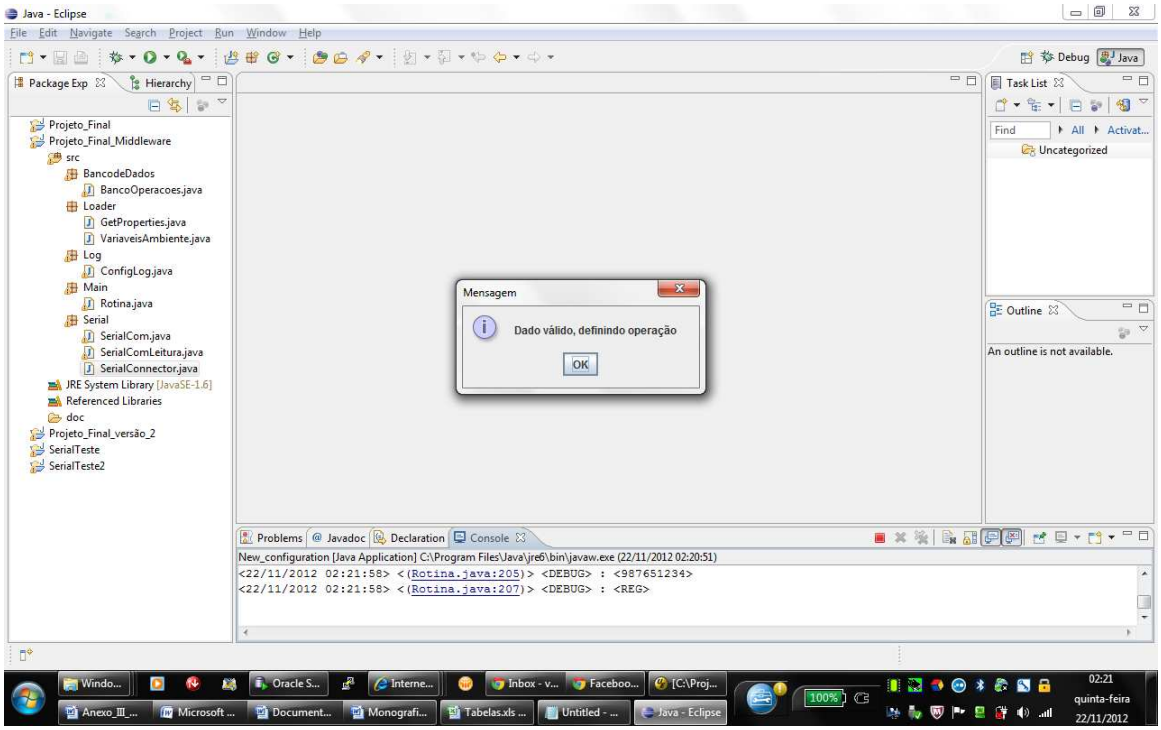

Figura E.8– Dado Válido
Fonte: Autor

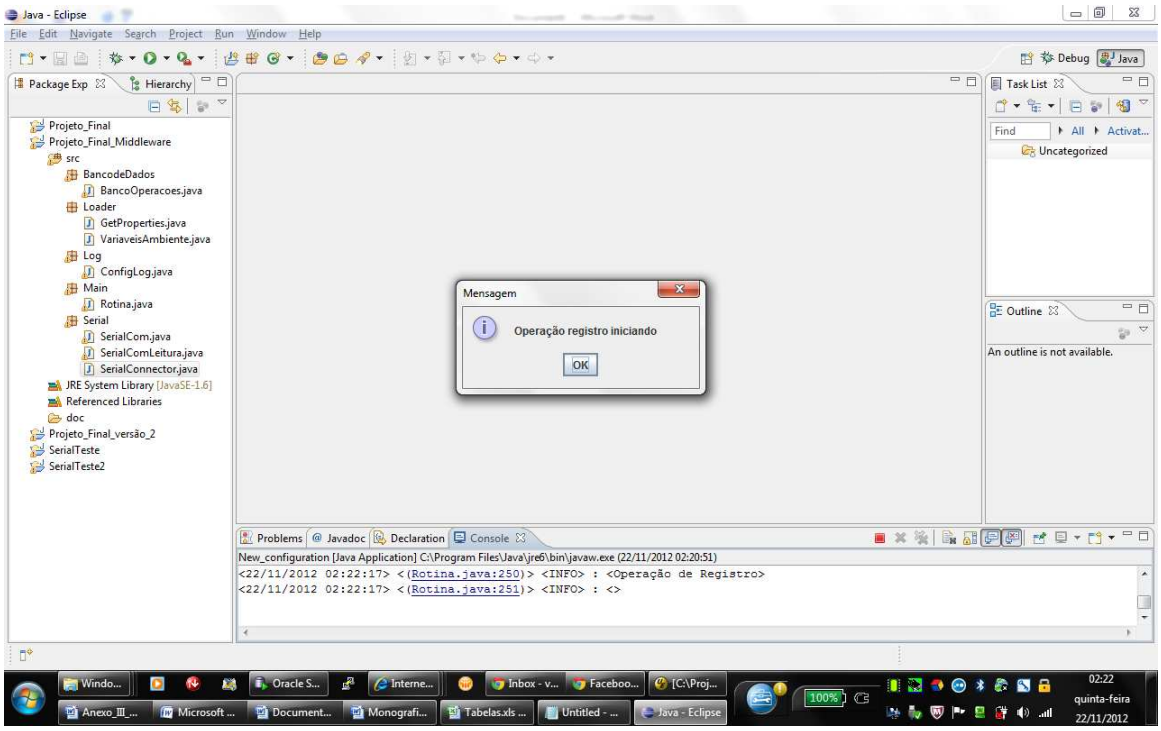

Figura E.9– Operação Registro
Fonte: Autor

Passo 5 - Acrescentar dado adicional;

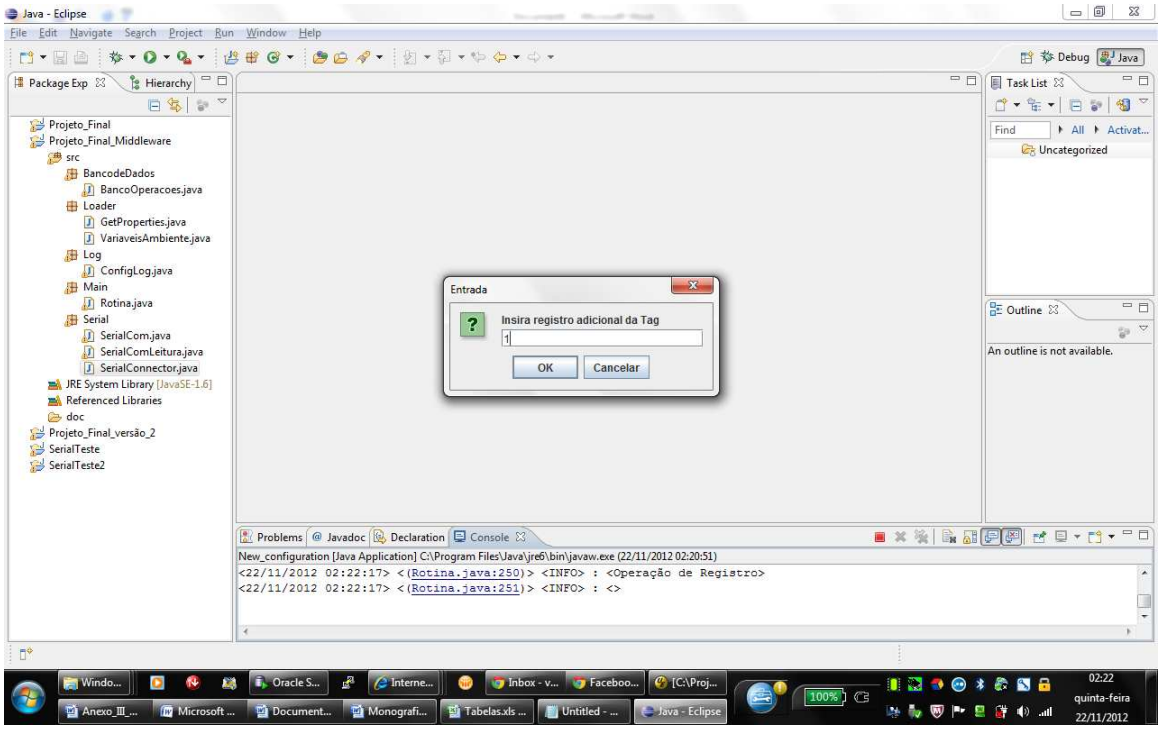

Figura E.10– Dado Adicional
Fonte: Autor

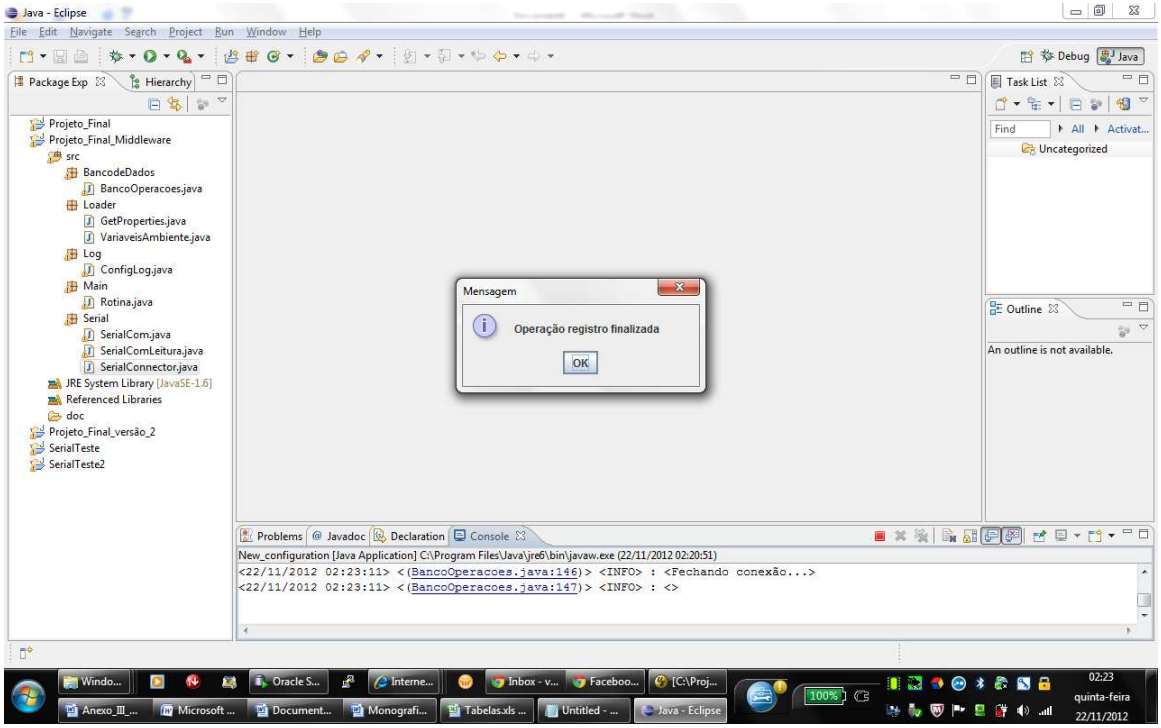

Figura E.11– Final Operação
Fonte: Autor

Passo 6 - Pesquisar dados no banco.

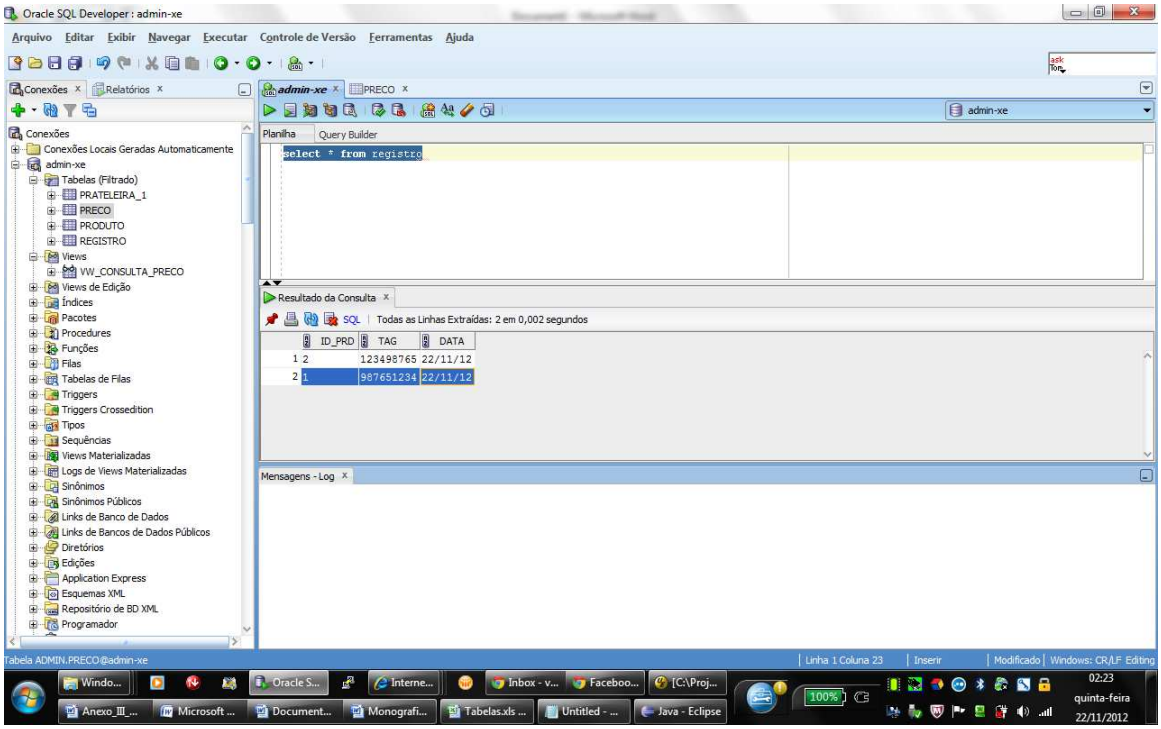

Figura E.12– Pesquisar
Fonte: Autor

Log completo do cenário:

<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: DEBUG>
<22/11/2012 02:20:54> <(Rotina.java:36)> <INFO> : <Nível de LOG setado para: DEBUG>
<22/11/2012 02:20:54> <(Rotina.java:37)> <INFO> : <>
<22/11/2012 02:20:54> <(Rotina.java:39)> <INFO> : <>
<22/11/2012 02:20:54> <(Rotina.java:40)> <INFO> : <Setando variáveis iniciais do módulo...>
<22/11/2012 02:20:54> <(Rotina.java:41)> <INFO> : <>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: oracle.jdbc.driver.OracleDriver>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: VW_CONSULTA_PRECO>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRATELEIRA_1>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: REGISTRO>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRECO>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: NOME>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 10000>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 9600>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: COM1>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: USB>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 02:20:54> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: ID_PRD>

<22/11/2012 02:20:54> <(Rotina.java:59)> <DEBUG> : <Driver: oracle.jdbc.driver.OracleDriver>
<22/11/2012 02:20:54> <(Rotina.java:60)> <DEBUG> : <Path: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 02:20:54> <(Rotina.java:61)> <DEBUG> : <BaudRate: 9600>
<22/11/2012 02:20:54> <(Rotina.java:62)> <DEBUG> : <SerialPorta: COM1>
<22/11/2012 02:20:54> <(Rotina.java:63)> <DEBUG> : <ModoEntrada: USB>
<22/11/2012 02:20:54> <(Rotina.java:64)> <DEBUG> : <SleepTime: 10000>
<22/11/2012 02:20:54> <(Rotina.java:65)> <DEBUG> : <Tabela Consulta: VW_CONSULTA_PRECO>
<22/11/2012 02:20:54> <(Rotina.java:66)> <DEBUG> : <Tabela Insert: PRATELEIRA_1>
<22/11/2012 02:20:54> <(Rotina.java:67)> <DEBUG> : <Tabela Registro: REGISTRO>
<22/11/2012 02:20:54> <(Rotina.java:68)> <DEBUG> : <Campo da tabela: PRECO>
<22/11/2012 02:20:54> <(Rotina.java:69)> <DEBUG> : <Campo da tabela: NOME>
<22/11/2012 02:20:54> <(Rotina.java:70)> <DEBUG> : <Dados1tabela: null>
<22/11/2012 02:20:54> <(Rotina.java:71)> <DEBUG> : <Dados2tabela: null>
<22/11/2012 02:20:54> <(Rotina.java:72)> <DEBUG> : <CampoPesquisa: TAG>
<22/11/2012 02:20:54> <(Rotina.java:73)> <DEBUG> : <ColunaInsert1: TAG>
<22/11/2012 02:20:54> <(Rotina.java:74)> <DEBUG> : <ColunaRegistro1: TAG>
<22/11/2012 02:20:54> <(Rotina.java:75)> <DEBUG> : <ColunaRegistro1: ID_PRD>
<22/11/2012 02:20:56> <(Rotina.java:107)> <DEBUG> : <Valor do modoentrada: USB>
<22/11/2012 02:20:58> <(Rotina.java:117)> <INFO> : <Aguardando input de dados...>
<22/11/2012 02:21:24> <(SerialConnector.java:188)> <DEBUG> : <Valor do str: 987651234;REG>
<22/11/2012 02:21:24> <(Rotina.java:124)> <DEBUG> : <Valor do entradados: 987651234;REG>
<22/11/2012 02:21:24> <(Rotina.java:176)> <DEBUG> : <Valor entradadados: 987651234;REG>
<22/11/2012 02:21:58> <(Rotina.java:186)> <DEBUG> : <Testetoken é: 987651234;REG>
<22/11/2012 02:21:58> <(Rotina.java:205)> <DEBUG> : <987651234>
<22/11/2012 02:21:58> <(Rotina.java:207)> <DEBUG> : <REG>
<22/11/2012 02:22:17> <(Rotina.java:249)> <INFO> : <>
<22/11/2012 02:22:17> <(Rotina.java:250)> <INFO> : <Operação de Registro>
<22/11/2012 02:22:17> <(Rotina.java:251)> <INFO> : <>
<22/11/2012 02:23:10> <(BancoOperacoes.java:112)> <INFO> : <>
<22/11/2012 02:23:10> <(BancoOperacoes.java:113)> <INFO> : <Setando variaveis para gravar no banco...>
<22/11/2012 02:23:10> <(BancoOperacoes.java:114)> <INFO> : <>
<22/11/2012 02:23:10> <(BancoOperacoes.java:115)> <INFO> : <Abrindo conexão...>
<22/11/2012 02:23:10> <(BancoOperacoes.java:116)> <INFO> : <>
<22/11/2012 02:23:11> <(BancoOperacoes.java:120)> <INFO> : <>
<22/11/2012 02:23:11> <(BancoOperacoes.java:121)> <INFO> : <populando dados do insert...>
<22/11/2012 02:23:11> <(BancoOperacoes.java:122)> <INFO> : <>
<22/11/2012 02:23:11> <(BancoOperacoes.java:125)> <DEBUG> : <SQL: INSERT INTO REGISTRO (TAG,ID_PRD, DATA) VALUES (? , ?, SYSDATE)>
<22/11/2012 02:23:11> <(BancoOperacoes.java:131)> <INFO> : <Executando insert...>
<22/11/2012 02:23:11> <(BancoOperacoes.java:132)> <INFO> : <>
<22/11/2012 02:23:11> <(BancoOperacoes.java:143)> <INFO> : <>
<22/11/2012 02:23:11> <(BancoOperacoes.java:144)> <INFO> : <Insert Executado com sucesso!>
<22/11/2012 02:23:11> <(BancoOperacoes.java:145)> <INFO> : <>
<22/11/2012 02:23:11> <(BancoOperacoes.java:146)> <INFO> : <Fechando conexão...>
<22/11/2012 02:23:11> <(BancoOperacoes.java:147)> <INFO> : <>

APÊNDICE F – Evidências Caso 5

Dados da Massa

Tag: 123498765

Dados Configuração BD

String conexão banco: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE
Tabela de Consulta: VW_CONSULTA_PRECO
Coluna Pesquisada: PRECO
Coluna Pesquisada adional: NOME
Campo Pesquisa: TAG

Dados Configuração Serial

Porta: COM1
Baudrate: 9600
Sleep: 10 segundos

Cenário: Enviar tag na porta serial com a operação CON

Passo 1 - Parametrizar arquivo middleware.properties

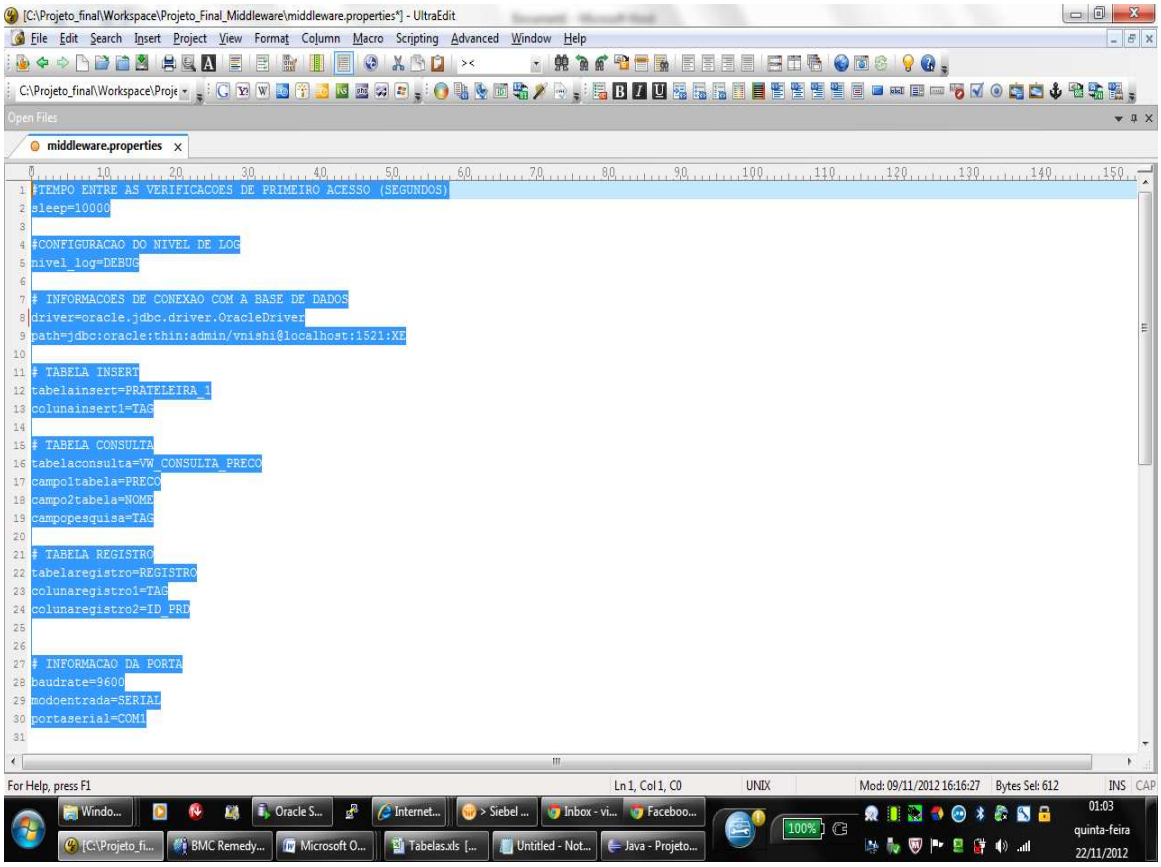

Figura F.1– Arquivo parametrizado
Fonte: Autor

Passo 2 - Validar que dado existe no banco

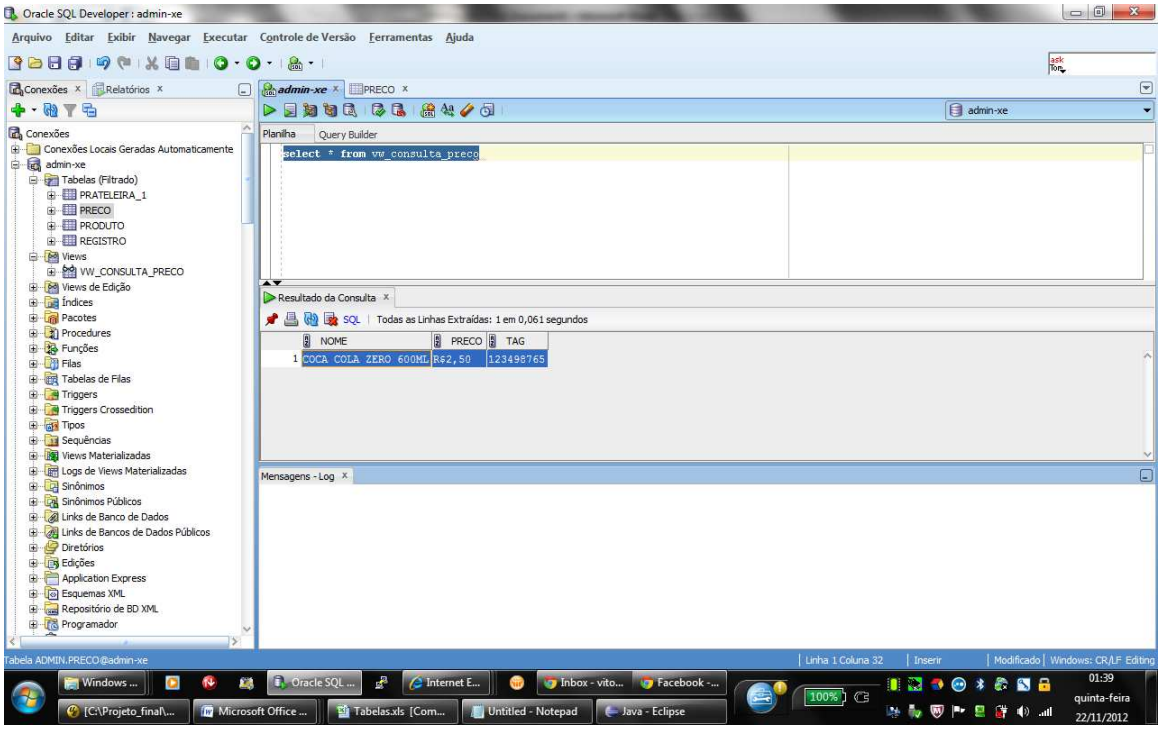

Figura F.2– Validação
Fonte: Autor

Passo 3 - Iniciar Middleware;

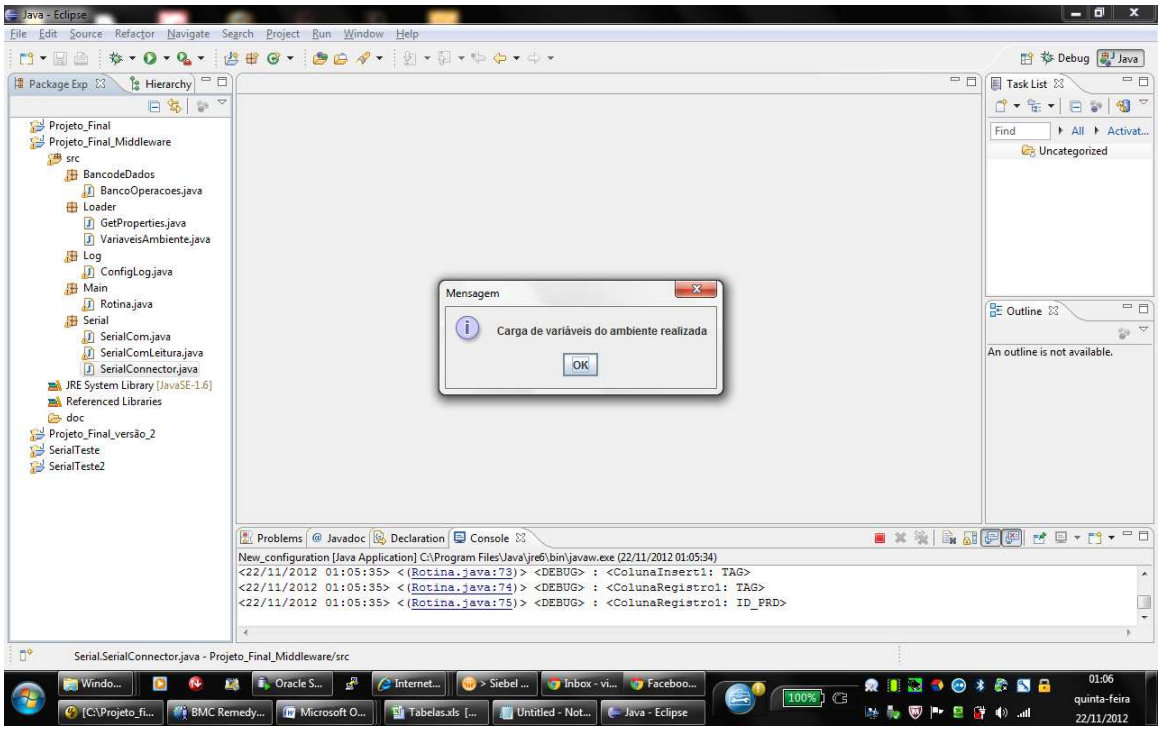

Figura F.3– Inicio Carga
Fonte: Autor

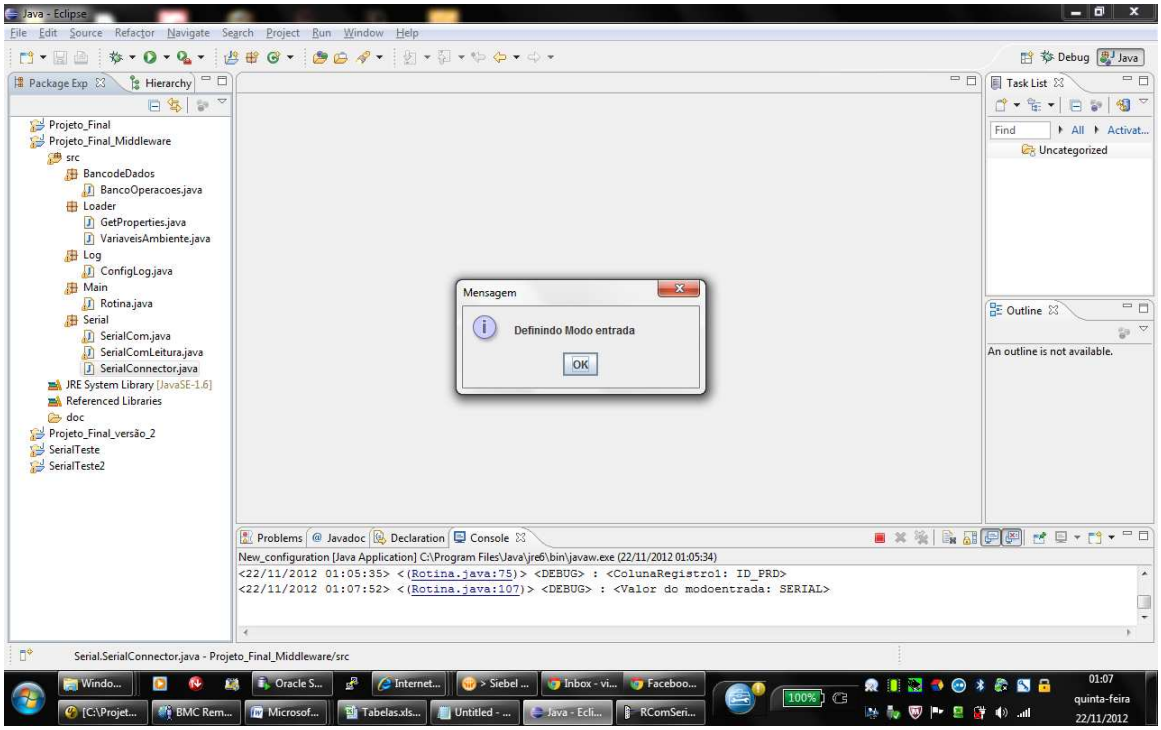

Figura F.4– Modo Entrada
Fonte: Autor

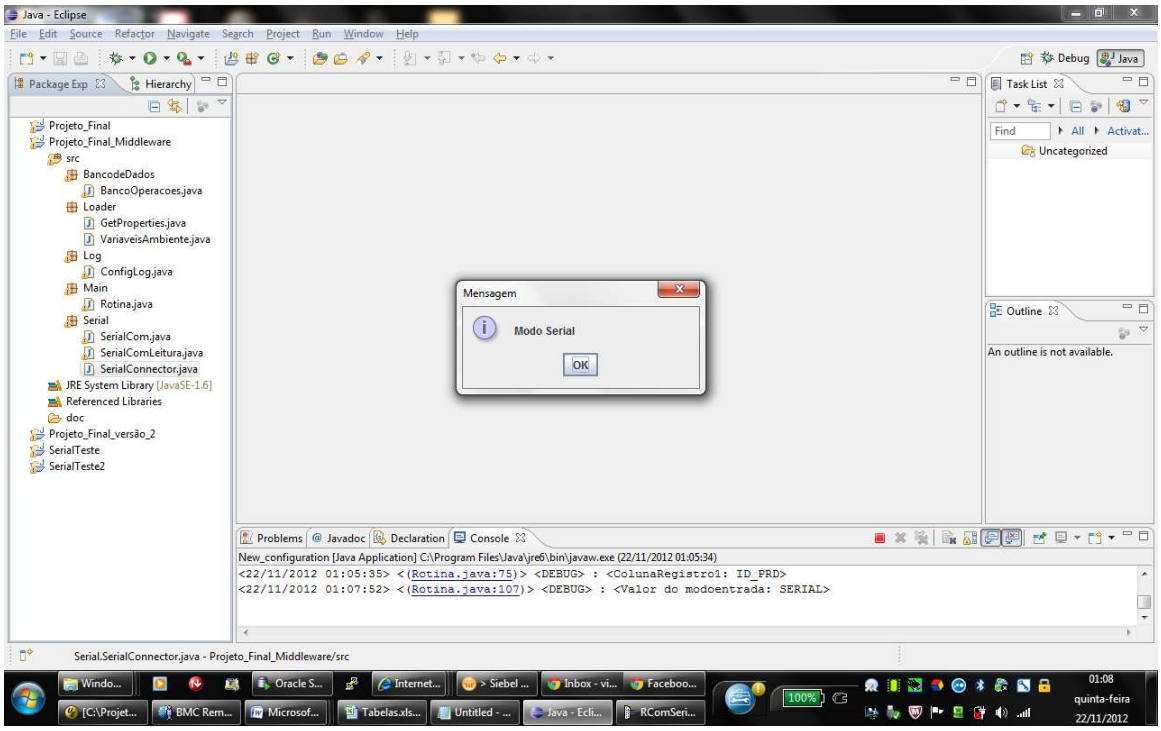

Figura F.5– Modo Serial
Fonte: Autor

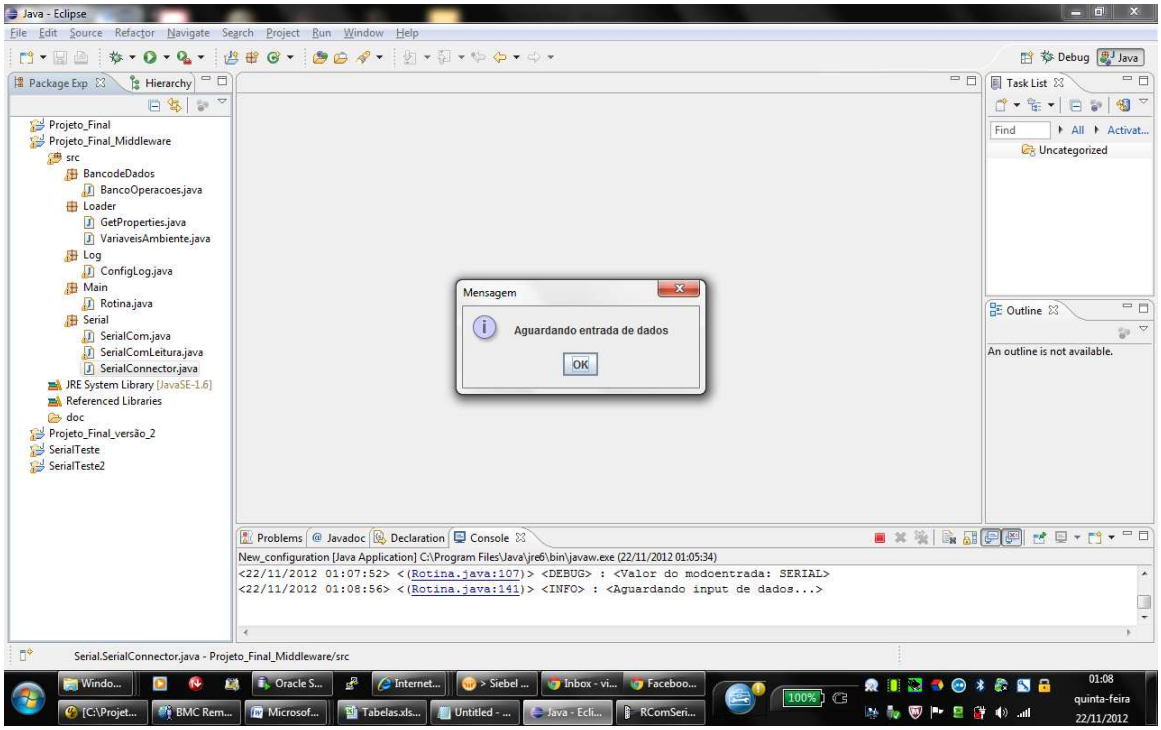

Figura F.6– Aguardando dados
Fonte: Autor

Passo 4 - Enviar dado;

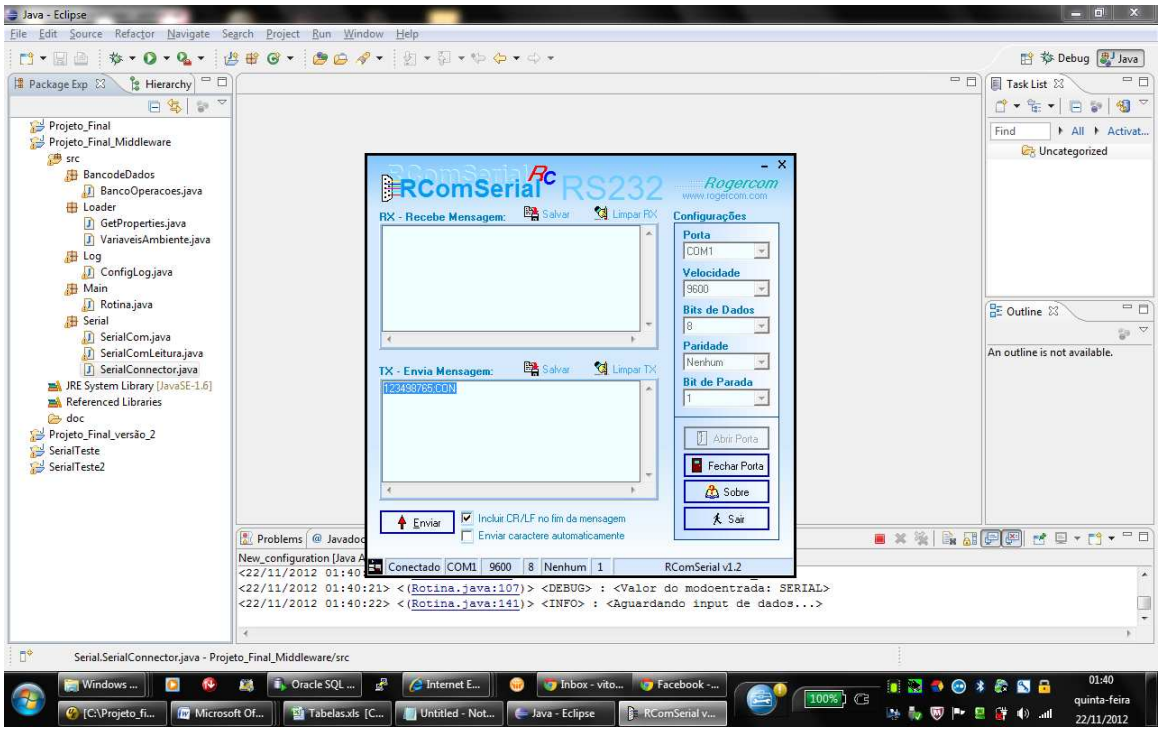

Figura F.7– Enviar Dados
Fonte: Autor

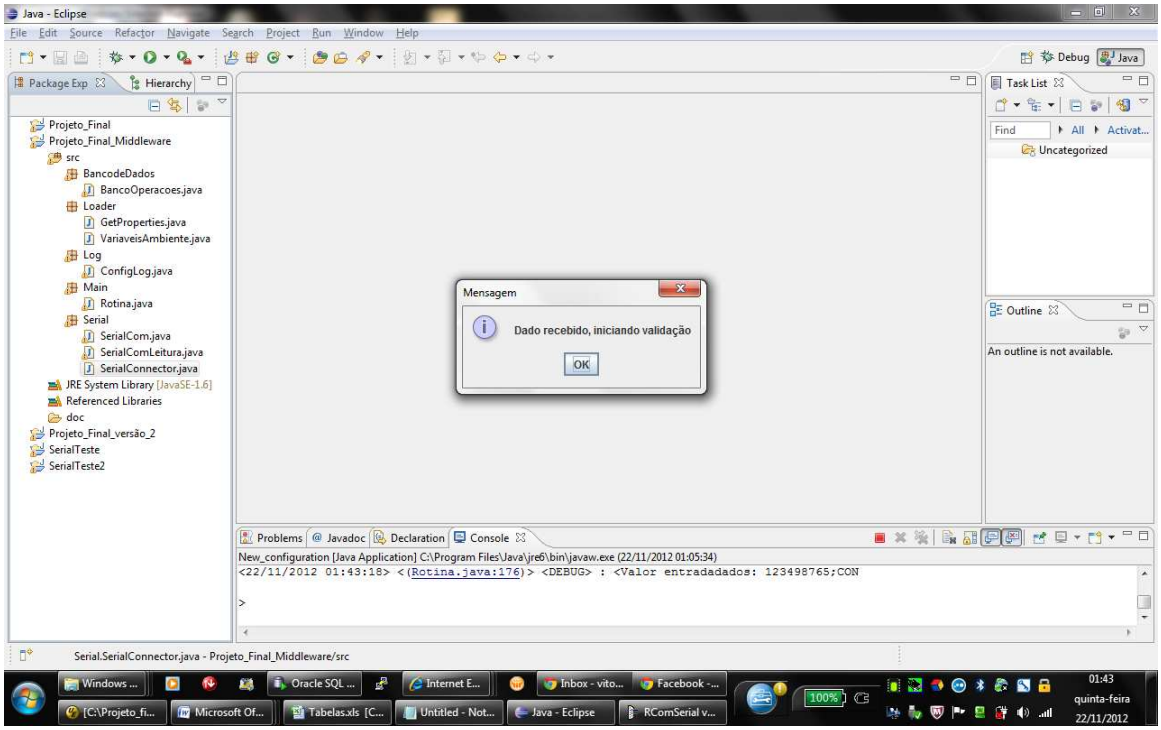

Figura F.8– Dados Recebidos
Fonte: Autor

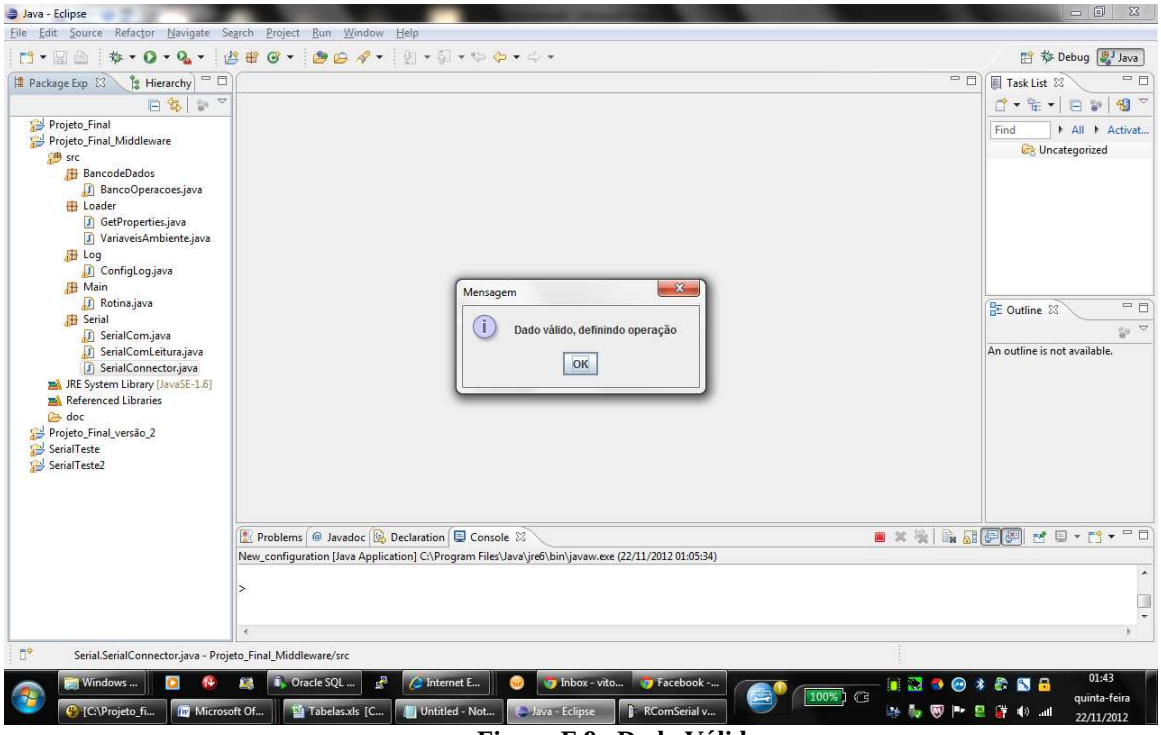

Figura F.9– Dado Válido
Fonte: Autor

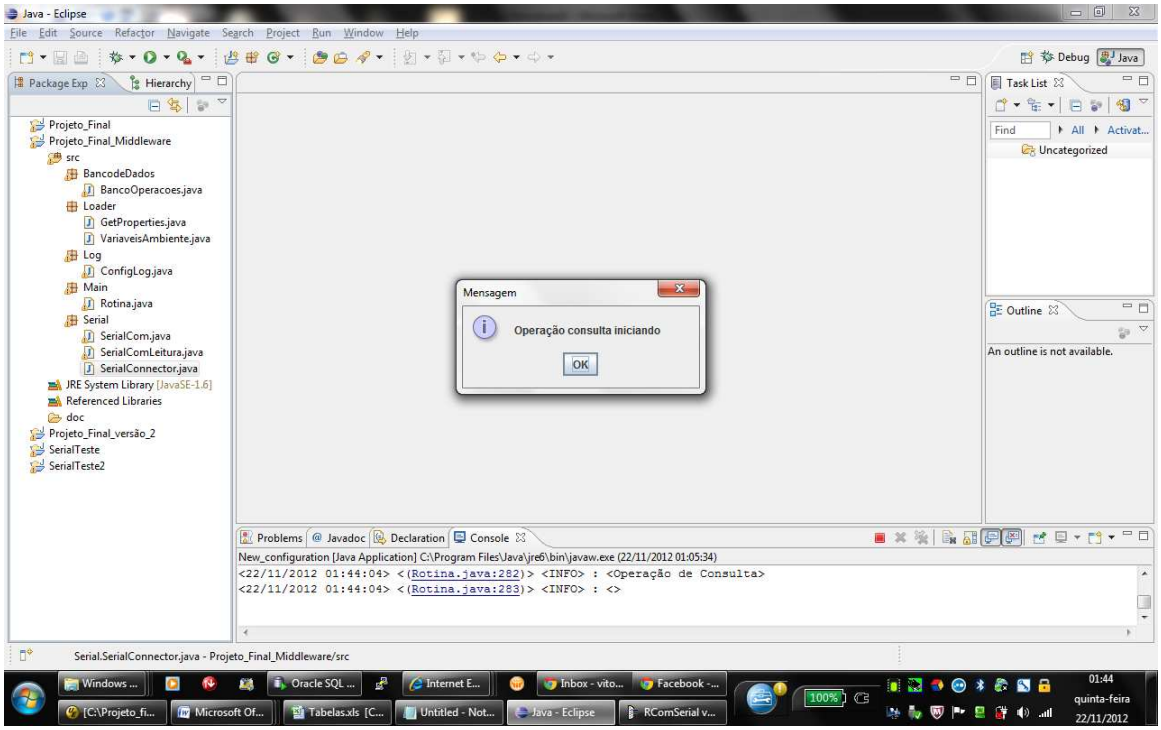

Figura F.10– Operação Consulta
Fonte: Autor

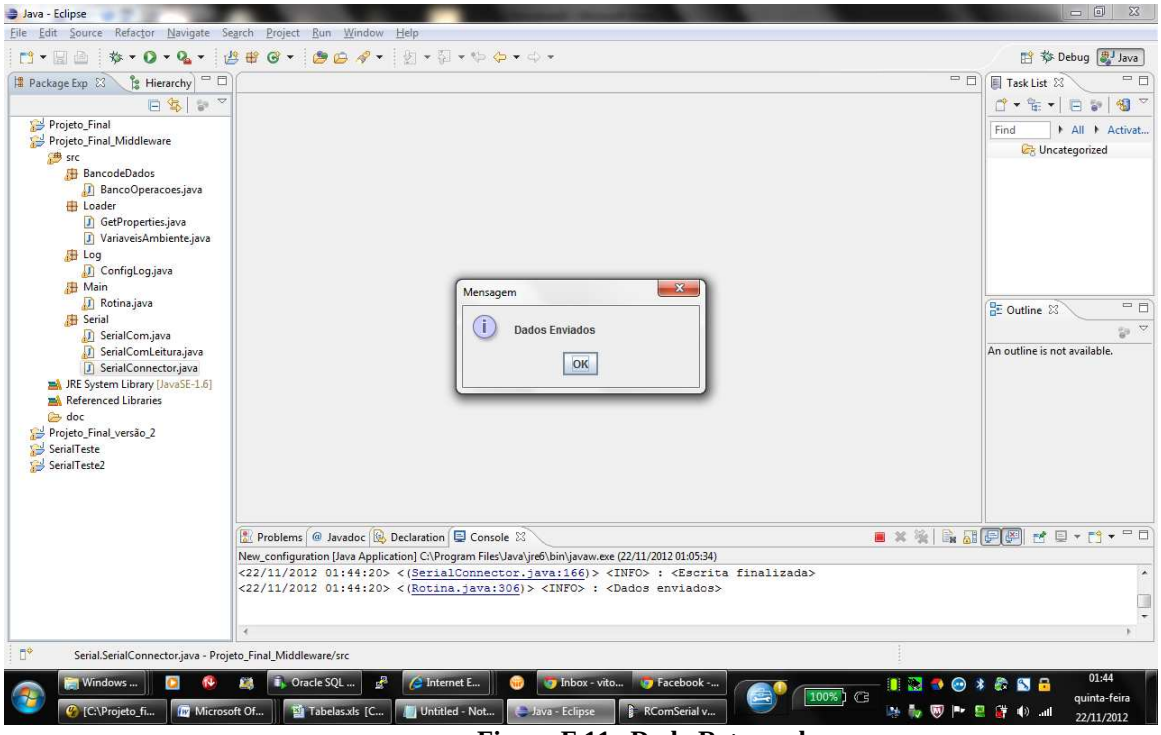

Figura F.11– Dado Retornados
Fonte: Autor

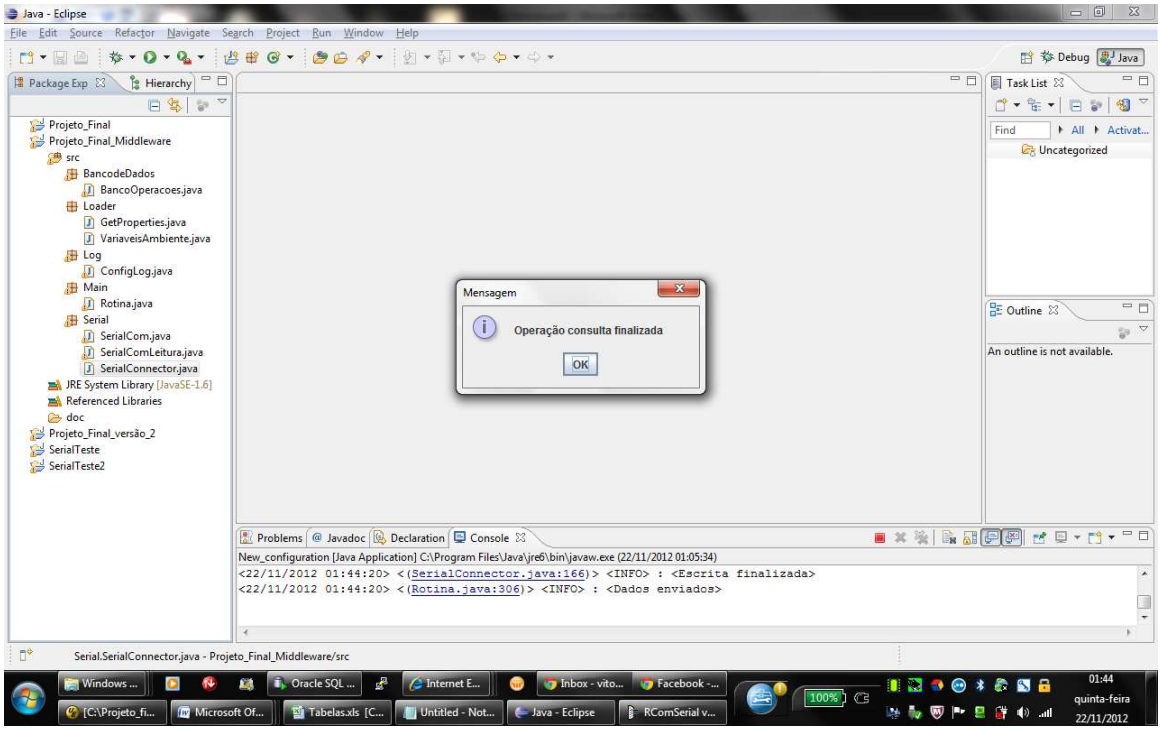

Figura F.12– Final Operação
Fonte: Autor

Passo 5 - Validar que dado retornou na serial

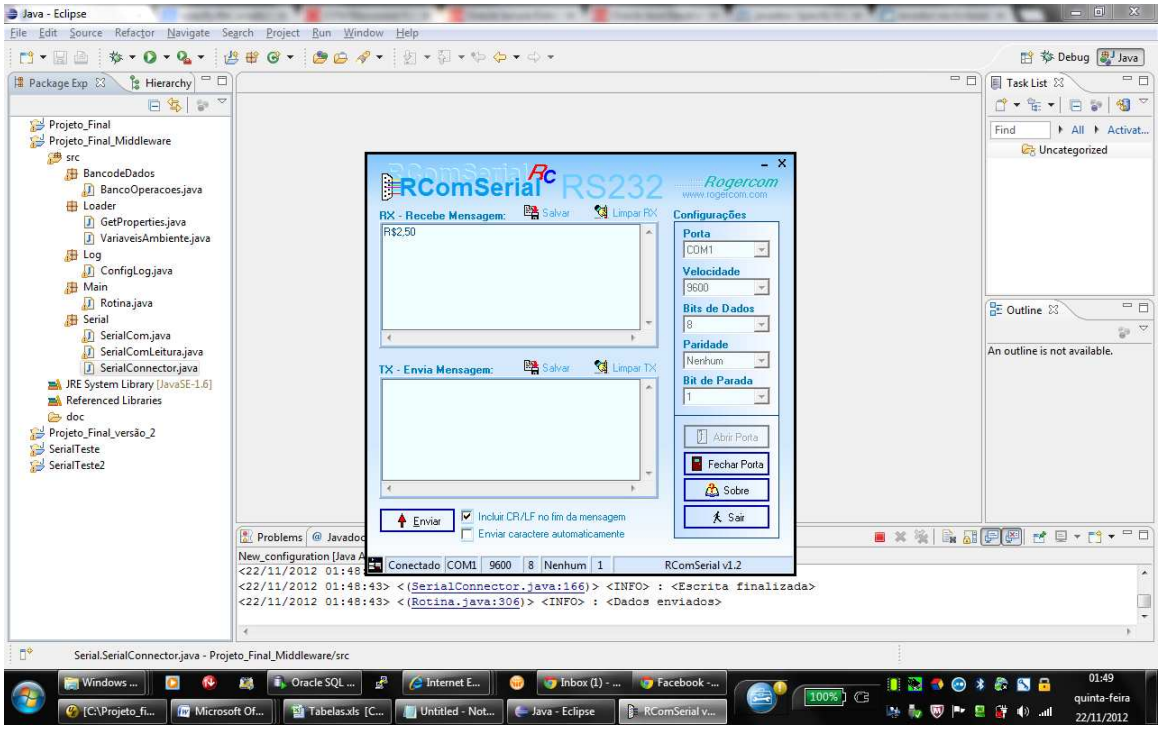

Figura F.13– Retorno Serial
Fonte: Autor

Log completo do cenário:

<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: DEBUG>
<22/11/2012 01:40:19> <(Rotina.java:36)> <INFO> : <Nível de LOG setado para: DEBUG>
<22/11/2012 01:40:19> <(Rotina.java:37)> <INFO> : <>
<22/11/2012 01:40:19> <(Rotina.java:39)> <INFO> : <>
<22/11/2012 01:40:19> <(Rotina.java:40)> <INFO> : <Setando variáveis iniciais do módulo...>
<22/11/2012 01:40:19> <(Rotina.java:41)> <INFO> : <>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: oracle.jdbc.driver.OracleDriver>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: VW_CONSULTA_PRECO>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRATELEIRA_1>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: REGISTRO>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRECO>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: NOME>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 10000>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 9600>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: COM1>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: SERIAL>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: ID_PRD>
<22/11/2012 01:40:19> <(Rotina.java:59)> <DEBUG> : <Driver: oracle.jdbc.driver.OracleDriver>
<22/11/2012 01:40:19> <(Rotina.java:60)> <DEBUG> : <Path: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 01:40:19> <(Rotina.java:61)> <DEBUG> : <BaudRate: 9600>
<22/11/2012 01:40:19> <(Rotina.java:62)> <DEBUG> : <SerialPorta: COM1>
<22/11/2012 01:40:19> <(Rotina.java:63)> <DEBUG> : <ModoEntrada: SERIAL>
<22/11/2012 01:40:19> <(Rotina.java:64)> <DEBUG> : <SleepTime: 10000>
<22/11/2012 01:40:19> <(Rotina.java:65)> <DEBUG> : <Tabela Consulta: VW_CONSULTA_PRECO>
<22/11/2012 01:40:19> <(Rotina.java:66)> <DEBUG> : <Tabela Insert: PRATELEIRA_1>
<22/11/2012 01:40:19> <(Rotina.java:67)> <DEBUG> : <Tabela Registro: REGISTRO>
<22/11/2012 01:40:19> <(Rotina.java:68)> <DEBUG> : <Campo da tabela: PRECO>
<22/11/2012 01:40:19> <(Rotina.java:69)> <DEBUG> : <Campo da tabela: NOME>
<22/11/2012 01:40:19> <(Rotina.java:70)> <DEBUG> : <Dados1tabela: null>
<22/11/2012 01:40:19> <(Rotina.java:71)> <DEBUG> : <Dados2tabela: null>
<22/11/2012 01:40:19> <(Rotina.java:72)> <DEBUG> : <CampoPesquisa: TAG>
<22/11/2012 01:40:19> <(Rotina.java:73)> <DEBUG> : <ColunaInsert1: TAG>
<22/11/2012 01:40:19> <(Rotina.java:74)> <DEBUG> : <ColunaRegistro1: TAG>
<22/11/2012 01:40:19> <(Rotina.java:75)> <DEBUG> : <ColunaRegistro1: ID_PRD>
<22/11/2012 01:40:21> <(Rotina.java:107)> <DEBUG> : <Valor do modoentrada: SERIAL>
<22/11/2012 01:40:22> <(Rotina.java:141)> <INFO> : <Aguardando input de dados...>
<22/11/2012 01:42:14> <(SerialConnector.java:25)> <INFO> : <Processo Leitura inicio>
<22/11/2012 01:42:14> <(SerialConnector.java:28)> <INFO> : <Leitura habilitada>
<22/11/2012 01:42:14> <(SerialConnector.java:33)> <INFO> : <Id porta obtido>
<22/11/2012 01:42:14> <(SerialConnector.java:38)> <INFO> : <Porta aberta>
<22/11/2012 01:42:14> <(SerialComLeitura.java:155)> <DEBUG> : <Valor entrada null>
<22/11/2012 01:42:14> <(SerialComLeitura.java:175)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@15e234c>

<22/11/2012 01:42:14> <(SerialComLeitura.java:196)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@15e234c>
<22/11/2012 01:42:14> <(SerialConnector.java:43)> <INFO> : <Poolling de leitura iniciado>
<22/11/2012 01:42:24> <(SerialConnector.java:57)> <INFO> : <Poolling de leitura finalizado>
<22/11/2012 01:42:24> <(SerialConnector.java:83)> <INFO> : <Leitura finalizada>
<22/11/2012 01:42:24> <(SerialConnector.java:93)> <DEBUG> : <Valor do buffer: null>
<22/11/2012 01:42:24> <(SerialConnector.java:94)> <INFO> : <<>>
<22/11/2012 01:42:24> <(Rotina.java:151)> <DEBUG> : <Valor do entradados: vazio>
<22/11/2012 01:42:37> <(SerialConnector.java:25)> <INFO> : <Processo Leitura inicio>
<22/11/2012 01:42:37> <(SerialConnector.java:28)> <INFO> : <Leitura habilitada>
<22/11/2012 01:42:37> <(SerialConnector.java:33)> <INFO> : <Id porta obtido>
<22/11/2012 01:42:37> <(SerialConnector.java:38)> <INFO> : <Porta aberta>
<22/11/2012 01:42:37> <(SerialComLeitura.java:155)> <DEBUG> : <Valor entrada null>
<22/11/2012 01:42:37> <(SerialComLeitura.java:175)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@1dc0e7a>
<22/11/2012 01:42:37> <(SerialComLeitura.java:196)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@1dc0e7a>
<22/11/2012 01:42:37> <(SerialConnector.java:43)> <INFO> : <Poolling de leitura iniciado>
<22/11/2012 01:42:47> <(SerialConnector.java:57)> <INFO> : <Poolling de leitura finalizado>
<22/11/2012 01:42:47> <(SerialConnector.java:83)> <INFO> : <Leitura finalizada>
<22/11/2012 01:42:47> <(SerialConnector.java:93)> <DEBUG> : <Valor do buffer: null>
<22/11/2012 01:42:47> <(SerialConnector.java:94)> <INFO> : <<>>
<22/11/2012 01:42:47> <(Rotina.java:151)> <DEBUG> : <Valor do entradados: vazio>
<22/11/2012 01:43:08> <(SerialConnector.java:25)> <INFO> : <Processo Leitura inicio>
<22/11/2012 01:43:08> <(SerialConnector.java:28)> <INFO> : <Leitura habilitada>
<22/11/2012 01:43:08> <(SerialConnector.java:33)> <INFO> : <Id porta obtido>
<22/11/2012 01:43:08> <(SerialConnector.java:38)> <INFO> : <Porta aberta>
<22/11/2012 01:43:08> <(SerialComLeitura.java:155)> <DEBUG> : <Valor entrada null>
<22/11/2012 01:43:08> <(SerialComLeitura.java:175)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@13e0aba>
<22/11/2012 01:43:08> <(SerialComLeitura.java:196)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@13e0aba>
<22/11/2012 01:43:08> <(SerialConnector.java:43)> <INFO> : <Poolling de leitura iniciado>
<22/11/2012 01:43:15> <(SerialComLeitura.java:286)> <DEBUG> : <Valor do bufferLeitura: >
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 49>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 49>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 50>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 50>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 51>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 51>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 52>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 52>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 57>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 57>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 56>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 56>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 55>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 55>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 54>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 54>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 53>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 53>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 59>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 59>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 67>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 67>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 79>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 79>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 78>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 78>

<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 13>
<22/11/2012 01:43:15> <(SerialComLeitura.java:331)> <DEBUG> : <Valor do novoDado: 13>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 10>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 10>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: -1>
<22/11/2012 01:43:15> <(SerialComLeitura.java:350)> <DEBUG> : <Valor do buffer: 123498765;CON

>
<22/11/2012 01:43:18> <(SerialConnector.java:57)> <INFO> : <Poolling de leitura finalizado>
<22/11/2012 01:43:18> <(SerialConnector.java:83)> <INFO> : <Leitura finalizada>
<22/11/2012 01:43:18> <(SerialConnector.java:93)> <DEBUG> : <Valor do buffer: 123498765;CON

>
<22/11/2012 01:43:18> <(SerialConnector.java:94)> <INFO> : <<>>
<22/11/2012 01:43:18> <(Rotina.java:151)> <DEBUG> : <Valor do entradados: 123498765;CON

>
<22/11/2012 01:43:18> <(Rotina.java:176)> <DEBUG> : <Valor entradadados: 123498765;CON

>
<22/11/2012 01:43:47> <(Rotina.java:186)> <DEBUG> : <Testetoken é: 123498765;CON

>
<22/11/2012 01:43:47> <(Rotina.java:205)> <DEBUG> : <123498765>
<22/11/2012 01:43:47> <(Rotina.java:207)> <DEBUG> : <CON

>
<22/11/2012 01:44:04> <(Rotina.java:281)> <INFO> : <>
<22/11/2012 01:44:04> <(Rotina.java:282)> <INFO> : <Operação de Consulta>
<22/11/2012 01:44:04> <(Rotina.java:283)> <INFO> : <>
<22/11/2012 01:44:19> <(BancoOperacoes.java:171)> <INFO> : <>
<22/11/2012 01:44:19> <(BancoOperacoes.java:172)> <INFO> : <Setando variaveis para fazer a consulta no banco...>
<22/11/2012 01:44:19> <(BancoOperacoes.java:173)> <INFO> : <>
<22/11/2012 01:44:19> <(BancoOperacoes.java:174)> <INFO> : <Abrindo conexão...>
<22/11/2012 01:44:19> <(BancoOperacoes.java:175)> <INFO> : <>
<22/11/2012 01:44:20> <(BancoOperacoes.java:179)> <INFO> : <>
<22/11/2012 01:44:20> <(BancoOperacoes.java:180)> <INFO> : <populando dados do select...>
<22/11/2012 01:44:20> <(BancoOperacoes.java:181)> <INFO> : <>
<22/11/2012 01:44:20> <(BancoOperacoes.java:184)> <DEBUG> : <sql: SELECT PRECO , NOME FROM VW_CONSULTA_PRECO WHERE TAG = '123498765' >
<22/11/2012 01:44:20> <(BancoOperacoes.java:186)> <INFO> : <Executando select...>
<22/11/2012 01:44:20> <(BancoOperacoes.java:187)> <INFO> : <>
<22/11/2012 01:44:20> <(BancoOperacoes.java:212)> <DEBUG> : <campos: [PRECO, NOME]>
<22/11/2012 01:44:20> <(BancoOperacoes.java:213)> <DEBUG> : <dados: [R$2,50, COCA COLA ZERO 600ML]>
<22/11/2012 01:44:20> <(BancoOperacoes.java:226)> <INFO> : <>
<22/11/2012 01:44:20> <(BancoOperacoes.java:227)> <INFO> : <Select Executado com sucesso!>
<22/11/2012 01:44:20> <(BancoOperacoes.java:228)> <INFO> : <>
<22/11/2012 01:44:20> <(BancoOperacoes.java:229)> <INFO> : <Fechando conexão...>
<22/11/2012 01:44:20> <(BancoOperacoes.java:230)> <INFO> : <>
<22/11/2012 01:44:20> <(Rotina.java:290)> <INFO> : <rs.size: 2>
<22/11/2012 01:44:20> <(Rotina.java:294)> <INFO> : <>
<22/11/2012 01:44:20> <(Rotina.java:295)> <INFO> : <Resultado da Consulta no banco:>
<22/11/2012 01:44:20> <(Rotina.java:296)> <INFO> : <DADO1: R$2,50>
<22/11/2012 01:44:20> <(Rotina.java:297)> <INFO> : <DADO2: COCA COLA ZERO 600ML>
<22/11/2012 01:44:20> <(Rotina.java:298)> <INFO> : <>
<22/11/2012 01:44:20> <(SerialConnector.java:112)> <INFO> : <Escrita iniciada>
<22/11/2012 01:44:20> <(SerialConnector.java:115)> <INFO> : <Escrita habilitada>
<22/11/2012 01:44:20> <(SerialConnector.java:120)> <INFO> : <Id porta obtido>

<22/11/2012 01:44:20> <(SerialConnector.java:125)> <INFO> : <Porta aberta>
<22/11/2012 01:44:20> <(SerialComLeitura.java:225)> <INFO> : <FLUXO OK!>
<22/11/2012 01:44:20> <(SerialComLeitura.java:235)> <INFO> : <Enviando um byte para COM1>
<22/11/2012 01:44:20> <(SerialComLeitura.java:237)> <DEBUG> : <Enviando : R$2,50>
<22/11/2012 01:44:20> <(SerialConnector.java:130)> <INFO> : <Dados enviados>
<22/11/2012 01:44:20> <(SerialConnector.java:156)> <INFO> : <Porta fechada>
<22/11/2012 01:44:20> <(SerialConnector.java:166)> <INFO> : <Escrita finalizada>
<22/11/2012 01:44:20> <(Rotina.java:306)> <INFO> : <Dados enviados>

APÊNDICE G – Evidências Caso 6

Dados da Massa

Tag: 987651234

Dados Configuração BD

String conexão banco: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE
Tabela de Consulta: VW_CONSULTA_PRECO
Coluna Pesquisada: PRECO
Coluna Pesquisada adional: NOME
Campo Pesquisa: TAG

Dados Configuração Serial

Porta: COM1
Baudrate: 9600
Sleep: 10 segundos

Cenário: Enviar tag na porta USB com a operação CON

Passo 1 - Parametrizar arquivo middleware.properties

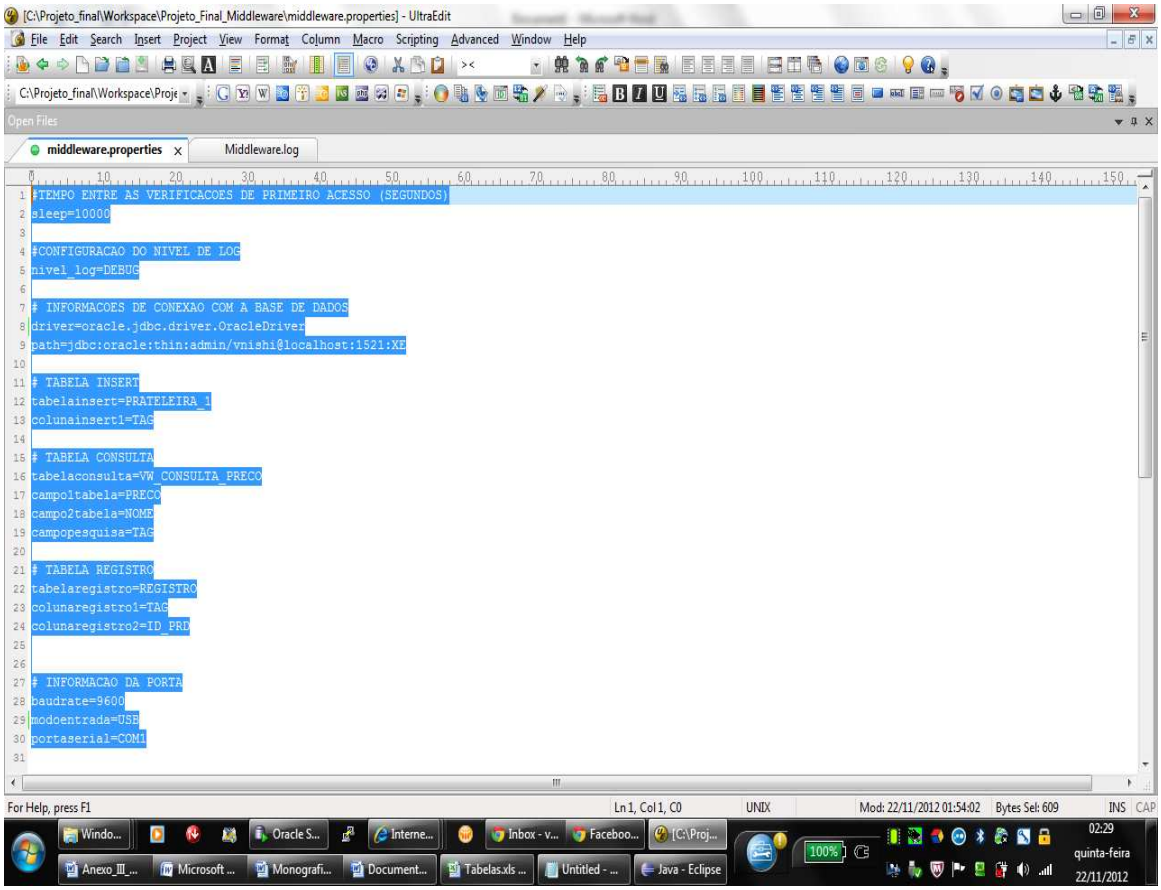

Figura G.1– Arquivo parametrizado
Fonte: Autor

Passo 2 - Validar que dado existe no banco

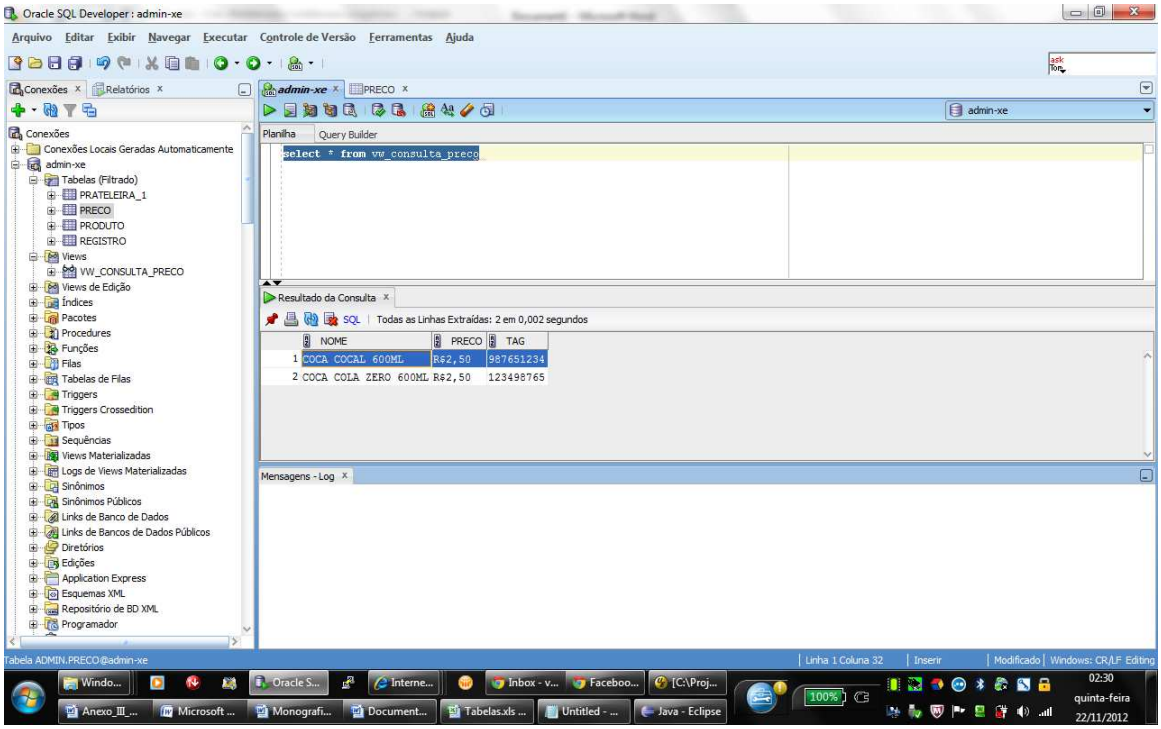

Figura G.2– Validação
Fonte: Autor

Passo 3 - Iniciar Middleware;

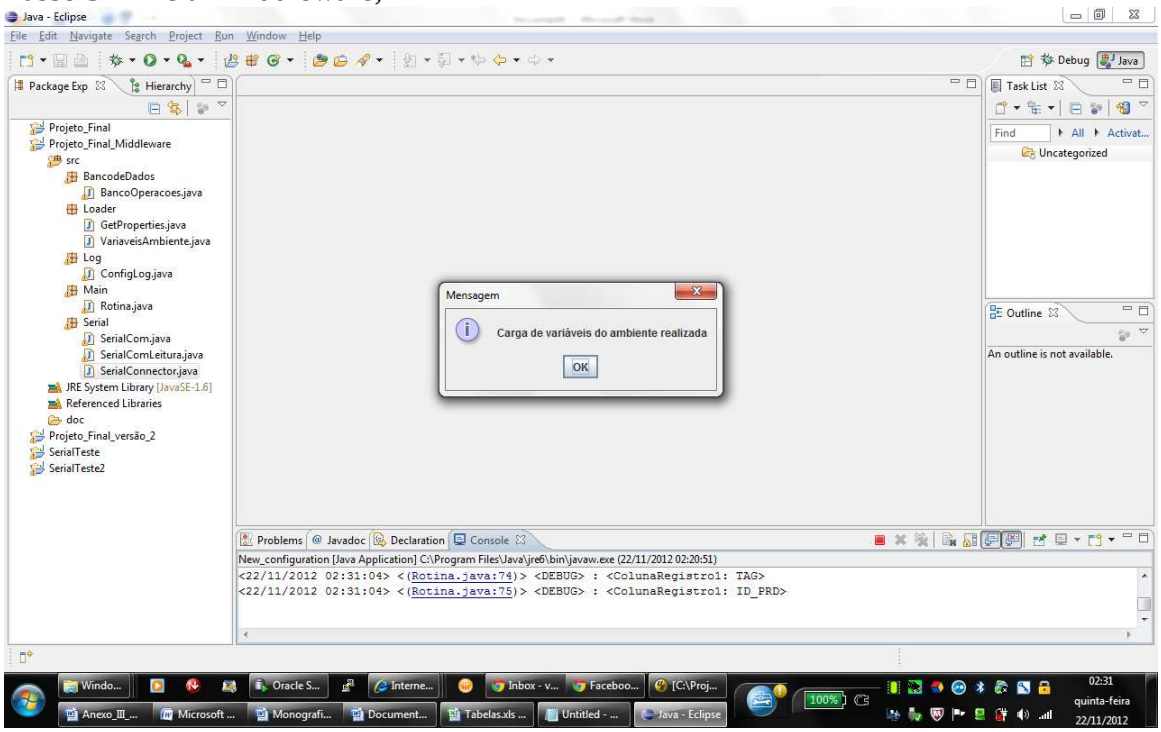

Figura G.3– Inicio Carga
Fonte: Autor

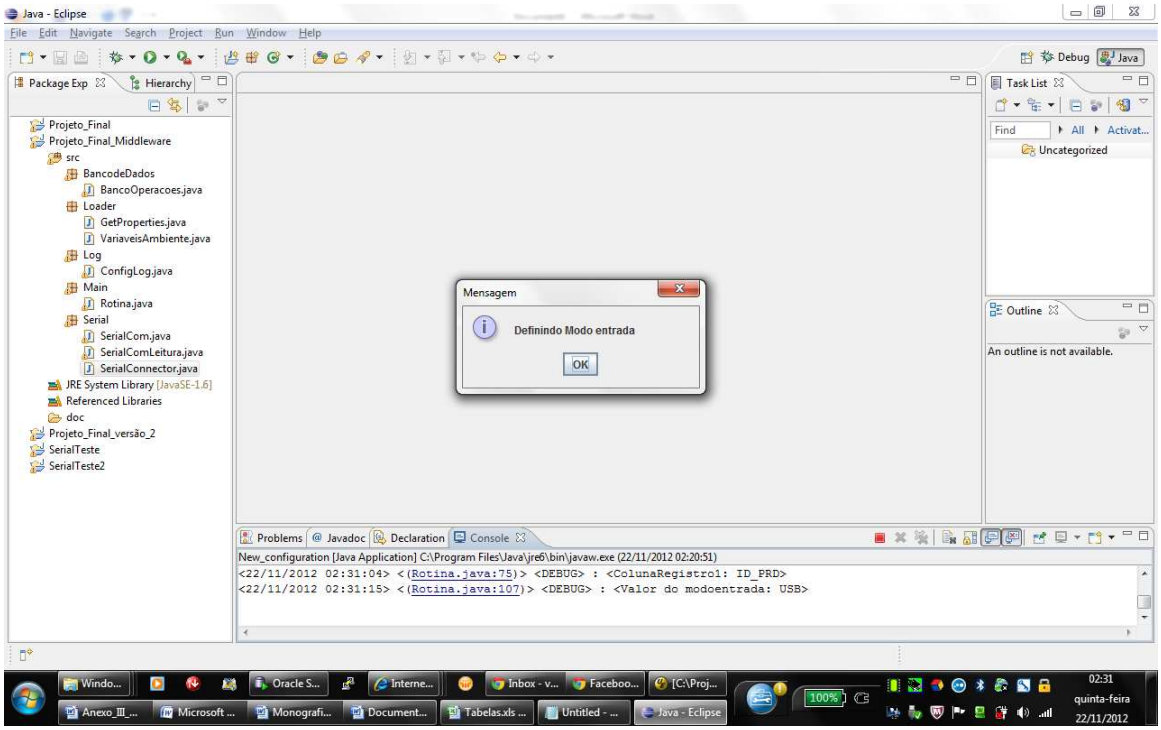

Figura G.4– Modo Entrada
Fonte: Autor

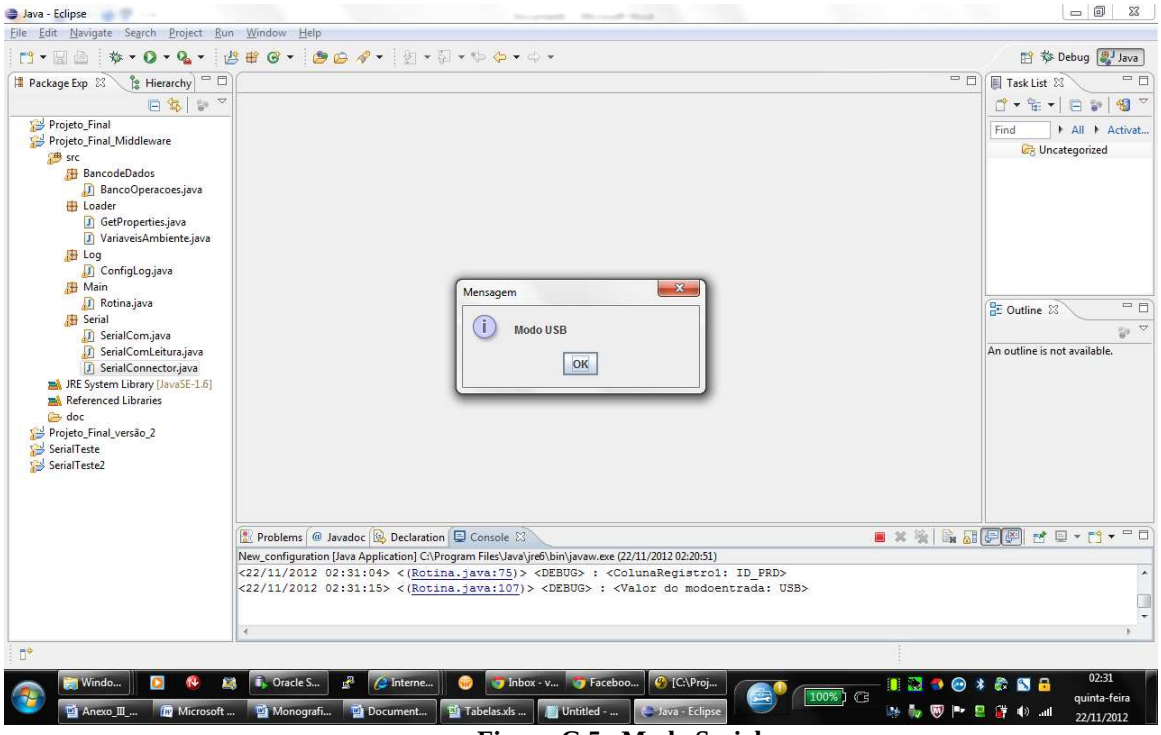

Figura G.5– Modo Serial
Fonte: Autor

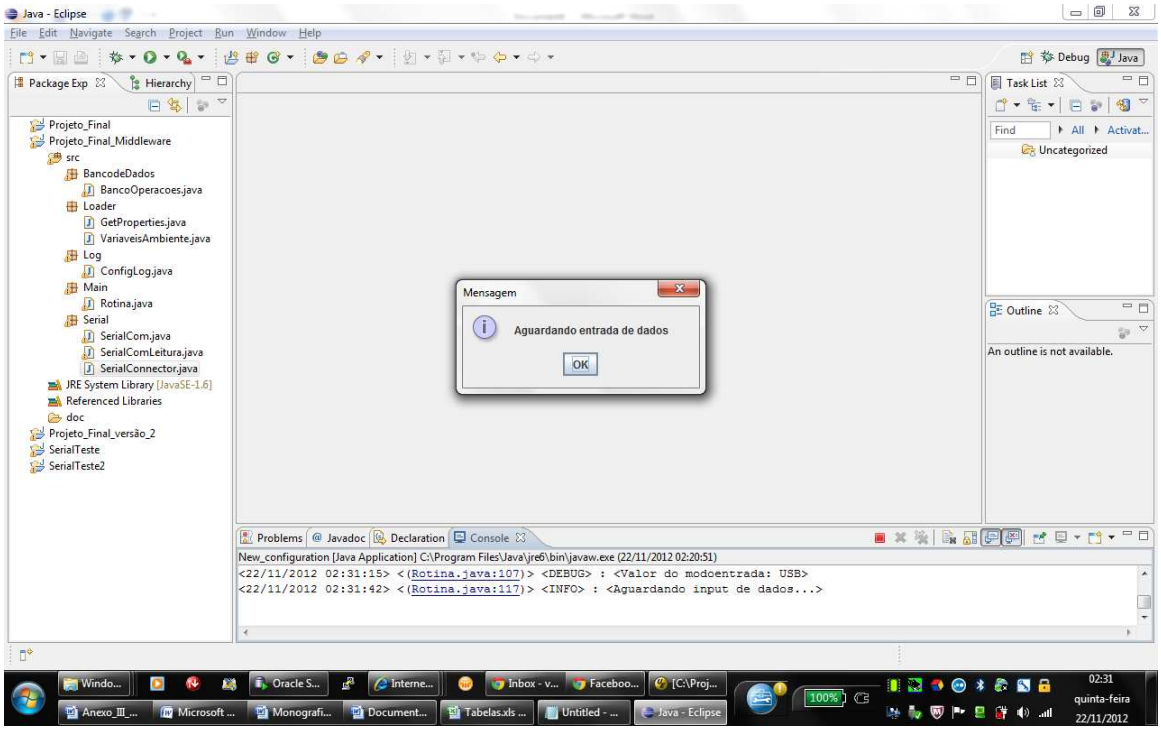

Figura G.6– Aguardando dados
Fonte: Autor

Passo 4 - Enviar dado;

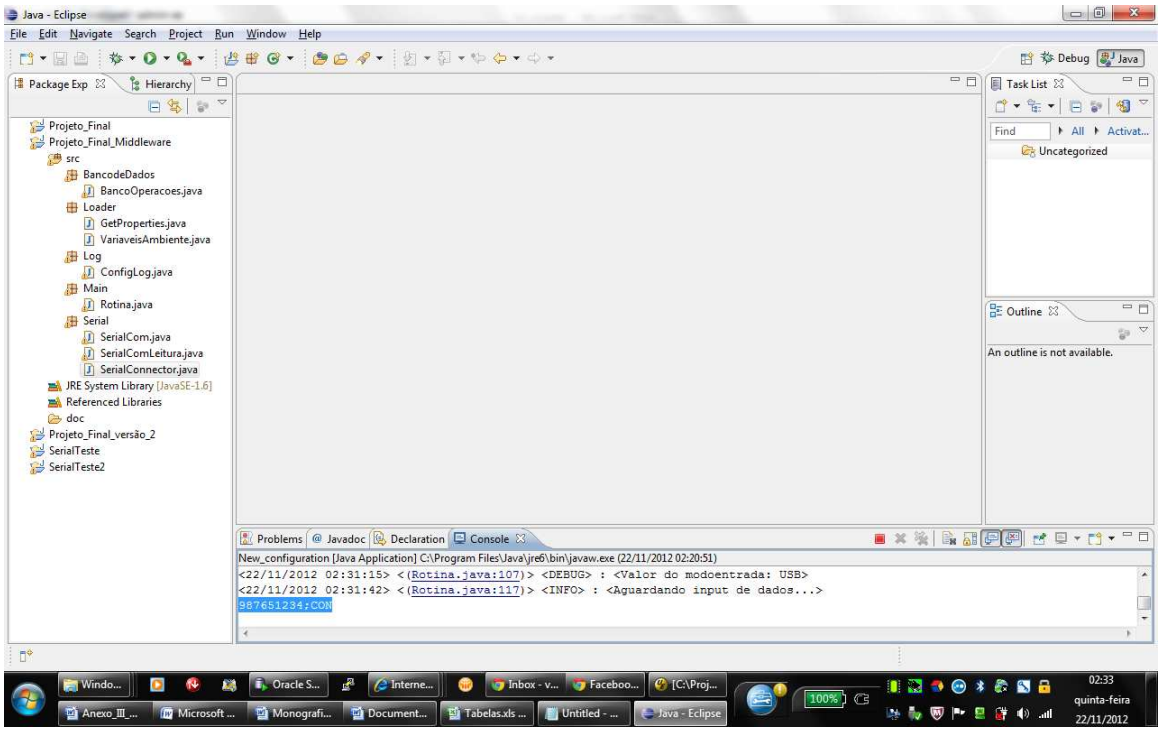

Figura G.7– Enviar Dados
Fonte: Autor

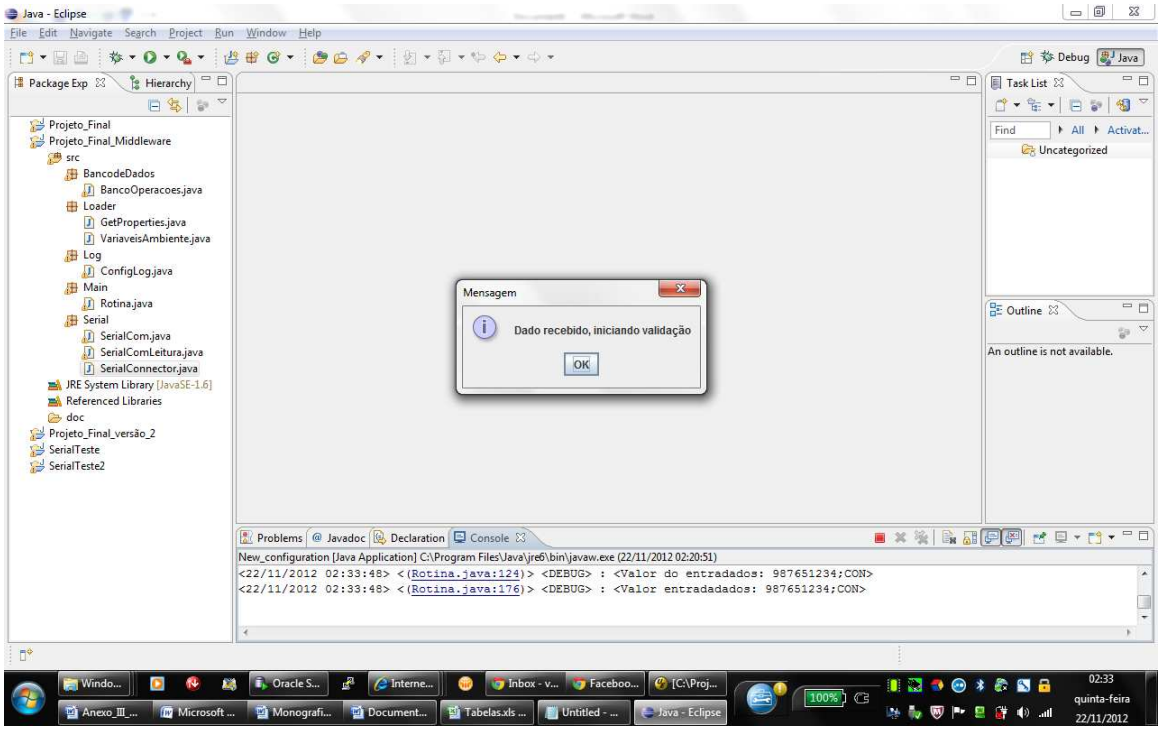

Figura G.8– Dados Recebidos
Fonte: Autor

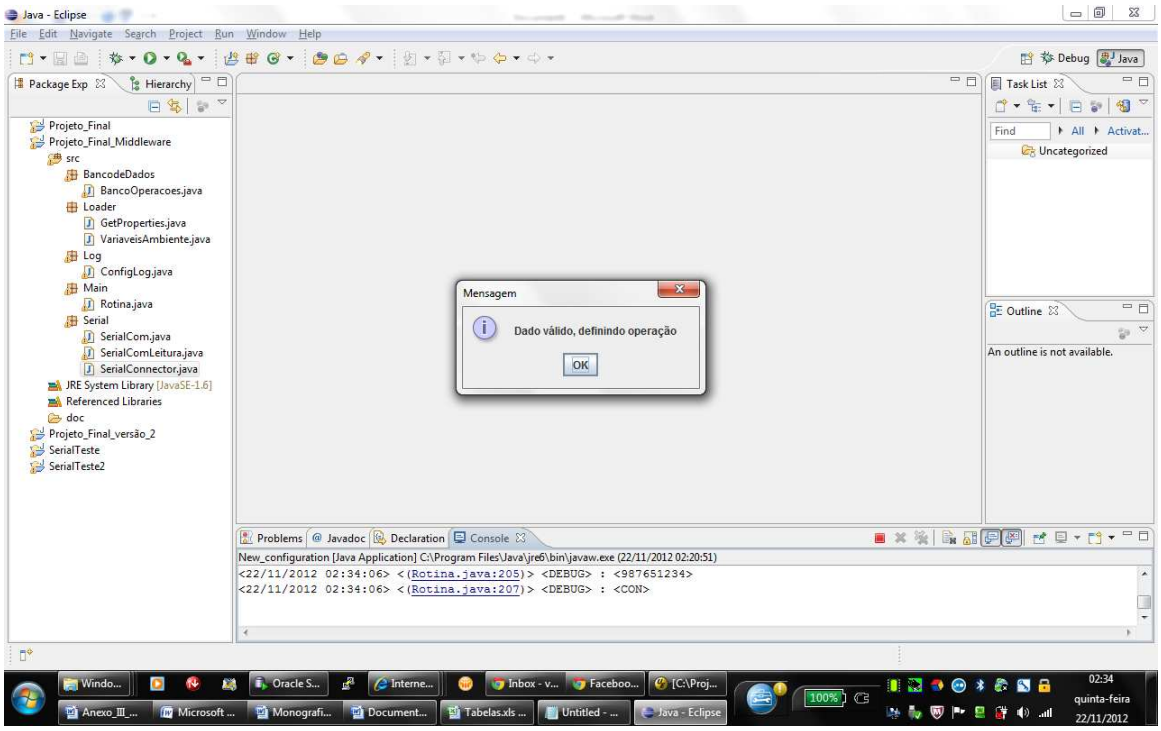

Figura G.9– Dado Válido
Fonte: Autor

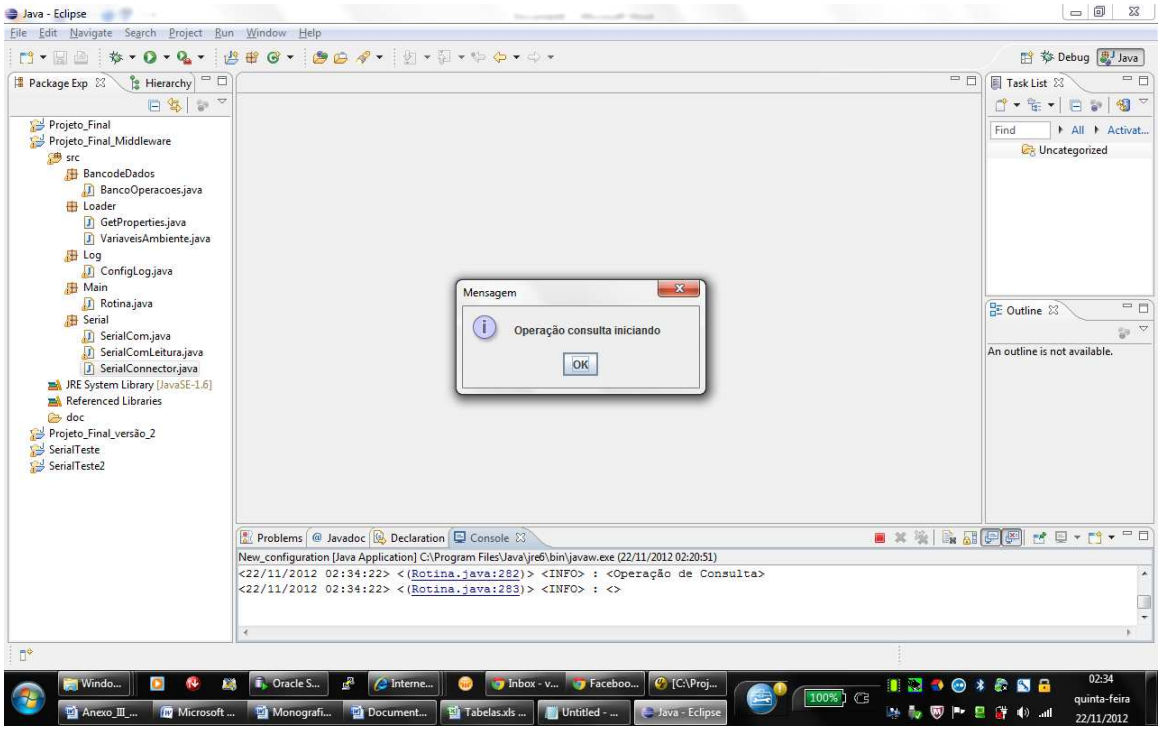

Figura G.10– Operação Consulta
Fonte: Autor

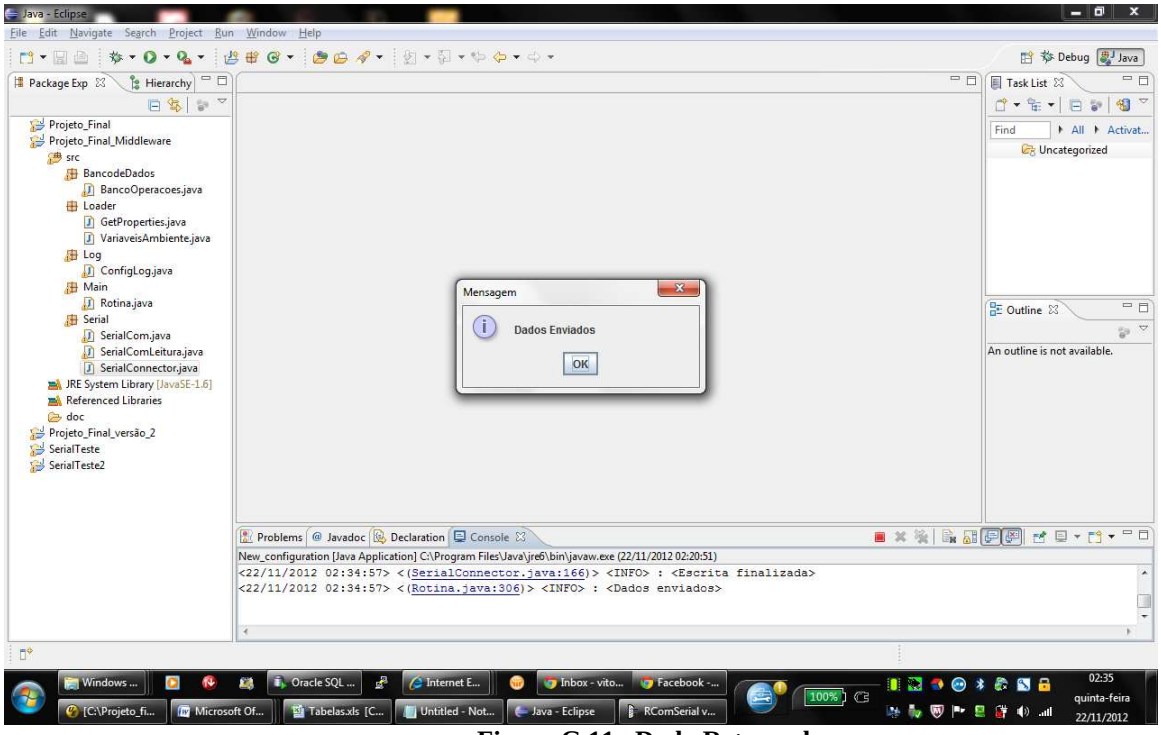

Figura G.11– Dado Retornados
Fonte: Autor

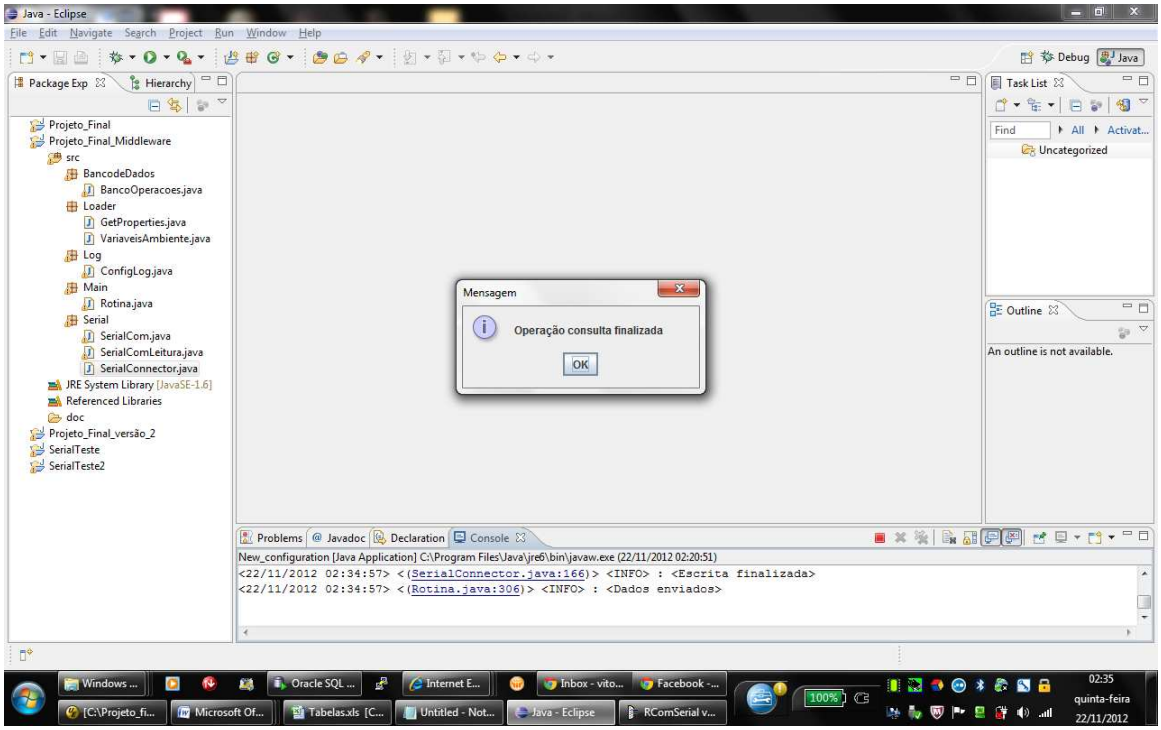

Figura G.12– Final Operação
Fonte: Autor

Passo 5 - Validar que dado retornou na serial

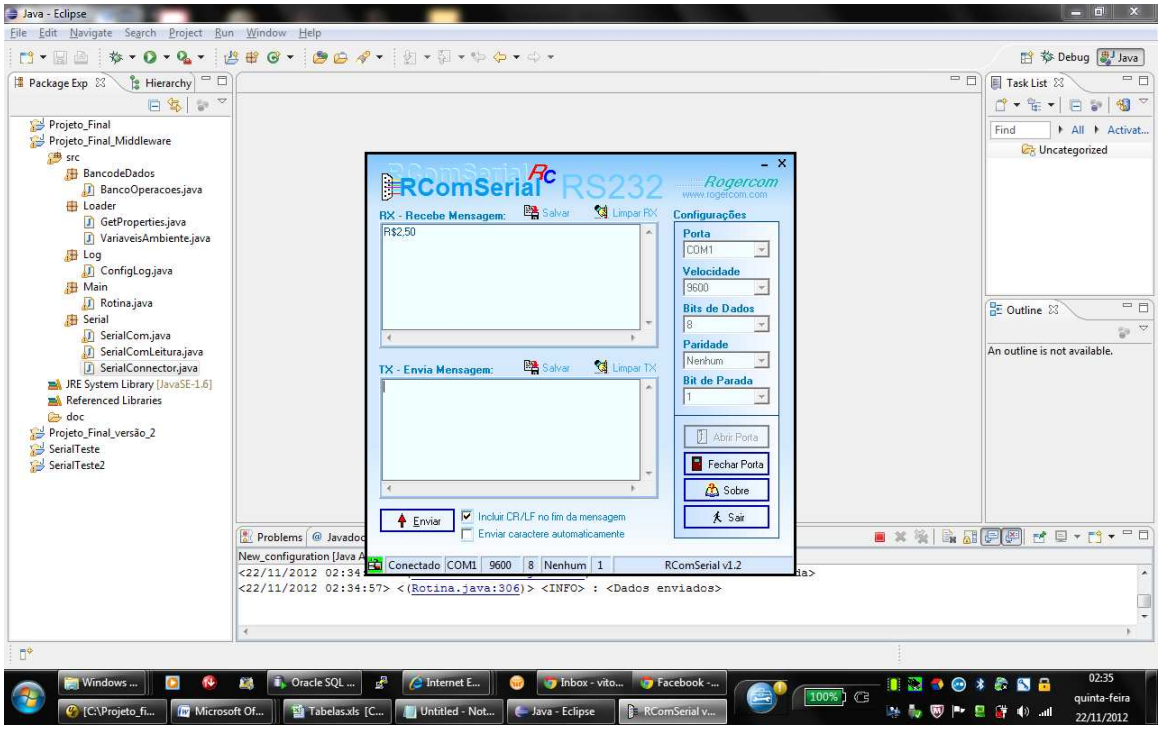

Figura G.13– Retorno Serial
Fonte: Autor

Log completo do cenário:

<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: DEBUG>
<22/11/2012 01:40:19> <(Rotina.java:36)> <INFO> : <Nível de LOG setado para: DEBUG>
<22/11/2012 01:40:19> <(Rotina.java:37)> <INFO> : <>
<22/11/2012 01:40:19> <(Rotina.java:39)> <INFO> : <>
<22/11/2012 01:40:19> <(Rotina.java:40)> <INFO> : <Setando variáveis iniciais do módulo...>
<22/11/2012 01:40:19> <(Rotina.java:41)> <INFO> : <>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: oracle.jdbc.driver.OracleDriver>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: VW_CONSULTA_PRECO>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRATELEIRA_1>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: REGISTRO>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: PRECO>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: NOME>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 10000>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: 9600>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: COM1>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: SERIAL>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: TAG>
<22/11/2012 01:40:19> <(GetProperties.java:60)> <DEBUG> : <Valor da variável propriedade: ID_PRD>
<22/11/2012 01:40:19> <(Rotina.java:59)> <DEBUG> : <Driver: oracle.jdbc.driver.OracleDriver>
<22/11/2012 01:40:19> <(Rotina.java:60)> <DEBUG> : <Path: jdbc:oracle:thin:admin/vnishi@localhost:1521:XE>
<22/11/2012 01:40:19> <(Rotina.java:61)> <DEBUG> : <BaudRate: 9600>
<22/11/2012 01:40:19> <(Rotina.java:62)> <DEBUG> : <SerialPorta: COM1>
<22/11/2012 01:40:19> <(Rotina.java:63)> <DEBUG> : <ModoEntrada: SERIAL>
<22/11/2012 01:40:19> <(Rotina.java:64)> <DEBUG> : <SleepTime: 10000>
<22/11/2012 01:40:19> <(Rotina.java:65)> <DEBUG> : <Tabela Consulta: VW_CONSULTA_PRECO>
<22/11/2012 01:40:19> <(Rotina.java:66)> <DEBUG> : <Tabela Insert: PRATELEIRA_1>
<22/11/2012 01:40:19> <(Rotina.java:67)> <DEBUG> : <Tabela Registro: REGISTRO>
<22/11/2012 01:40:19> <(Rotina.java:68)> <DEBUG> : <Campo da tabela: PRECO>
<22/11/2012 01:40:19> <(Rotina.java:69)> <DEBUG> : <Campo da tabela: NOME>
<22/11/2012 01:40:19> <(Rotina.java:70)> <DEBUG> : <Dados1tabela: null>
<22/11/2012 01:40:19> <(Rotina.java:71)> <DEBUG> : <Dados2tabela: null>
<22/11/2012 01:40:19> <(Rotina.java:72)> <DEBUG> : <CampoPesquisa: TAG>
<22/11/2012 01:40:19> <(Rotina.java:73)> <DEBUG> : <ColunaInsert1: TAG>
<22/11/2012 01:40:19> <(Rotina.java:74)> <DEBUG> : <ColunaRegistro1: TAG>
<22/11/2012 01:40:19> <(Rotina.java:75)> <DEBUG> : <ColunaRegistro1: ID_PRD>
<22/11/2012 01:40:21> <(Rotina.java:107)> <DEBUG> : <Valor do modoentrada: SERIAL>
<22/11/2012 01:40:22> <(Rotina.java:141)> <INFO> : <Aguardando input de dados...>
<22/11/2012 01:42:14> <(SerialConnector.java:25)> <INFO> : <Processo Leitura inicio>
<22/11/2012 01:42:14> <(SerialConnector.java:28)> <INFO> : <Leitura habilitada>
<22/11/2012 01:42:14> <(SerialConnector.java:33)> <INFO> : <Id porta obtido>
<22/11/2012 01:42:14> <(SerialConnector.java:38)> <INFO> : <Porta aberta>
<22/11/2012 01:42:14> <(SerialComLeitura.java:155)> <DEBUG> : <Valor entrada null>
<22/11/2012 01:42:14> <(SerialComLeitura.java:175)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@15e234c>

<22/11/2012 01:42:14> <(SerialComLeitura.java:196)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@15e234c>
<22/11/2012 01:42:14> <(SerialConnector.java:43)> <INFO> : <Poolling de leitura iniciado>
<22/11/2012 01:42:24> <(SerialConnector.java:57)> <INFO> : <Poolling de leitura finalizado>
<22/11/2012 01:42:24> <(SerialConnector.java:83)> <INFO> : <Leitura finalizada>
<22/11/2012 01:42:24> <(SerialConnector.java:93)> <DEBUG> : <Valor do buffer: null>
<22/11/2012 01:42:24> <(SerialConnector.java:94)> <INFO> : <<>>
<22/11/2012 01:42:24> <(Rotina.java:151)> <DEBUG> : <Valor do entradados: vazio>
<22/11/2012 01:42:37> <(SerialConnector.java:25)> <INFO> : <Processo Leitura inicio>
<22/11/2012 01:42:37> <(SerialConnector.java:28)> <INFO> : <Leitura habilitada>
<22/11/2012 01:42:37> <(SerialConnector.java:33)> <INFO> : <Id porta obtido>
<22/11/2012 01:42:37> <(SerialConnector.java:38)> <INFO> : <Porta aberta>
<22/11/2012 01:42:37> <(SerialComLeitura.java:155)> <DEBUG> : <Valor entrada null>
<22/11/2012 01:42:37> <(SerialComLeitura.java:175)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@1dc0e7a>
<22/11/2012 01:42:37> <(SerialComLeitura.java:196)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@1dc0e7a>
<22/11/2012 01:42:37> <(SerialConnector.java:43)> <INFO> : <Poolling de leitura iniciado>
<22/11/2012 01:42:47> <(SerialConnector.java:57)> <INFO> : <Poolling de leitura finalizado>
<22/11/2012 01:42:47> <(SerialConnector.java:83)> <INFO> : <Leitura finalizada>
<22/11/2012 01:42:47> <(SerialConnector.java:93)> <DEBUG> : <Valor do buffer: null>
<22/11/2012 01:42:47> <(SerialConnector.java:94)> <INFO> : <<>>
<22/11/2012 01:42:47> <(Rotina.java:151)> <DEBUG> : <Valor do entradados: vazio>
<22/11/2012 01:43:08> <(SerialConnector.java:25)> <INFO> : <Processo Leitura inicio>
<22/11/2012 01:43:08> <(SerialConnector.java:28)> <INFO> : <Leitura habilitada>
<22/11/2012 01:43:08> <(SerialConnector.java:33)> <INFO> : <Id porta obtido>
<22/11/2012 01:43:08> <(SerialConnector.java:38)> <INFO> : <Porta aberta>
<22/11/2012 01:43:08> <(SerialComLeitura.java:155)> <DEBUG> : <Valor entrada null>
<22/11/2012 01:43:08> <(SerialComLeitura.java:175)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@13e0aba>
<22/11/2012 01:43:08> <(SerialComLeitura.java:196)> <DEBUG> : <Valor da entrada: gnu.io.RXTXPort$SerialInputStream@13e0aba>
<22/11/2012 01:43:08> <(SerialConnector.java:43)> <INFO> : <Poolling de leitura iniciado>
<22/11/2012 01:43:15> <(SerialComLeitura.java:286)> <DEBUG> : <Valor do bufferLeitura: >
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 49>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 49>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 50>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 50>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 51>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 51>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 52>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 52>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 57>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 57>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 56>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 56>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 55>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 55>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 54>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 54>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 53>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 53>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 59>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 59>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 67>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 67>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 79>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 79>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 78>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 78>

<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 13>
<22/11/2012 01:43:15> <(SerialComLeitura.java:331)> <DEBUG> : <Valor do novoDado: 13>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: 10>
<22/11/2012 01:43:15> <(SerialComLeitura.java:336)> <DEBUG> : <Valor do novoDado: 10>
<22/11/2012 01:43:15> <(SerialComLeitura.java:319)> <DEBUG> : <Valor do novoDado: -1>
<22/11/2012 01:43:15> <(SerialComLeitura.java:350)> <DEBUG> : <Valor do buffer: 123498765;CON

>
<22/11/2012 01:43:18> <(SerialConnector.java:57)> <INFO> : <Poolling de leitura finalizado>
<22/11/2012 01:43:18> <(SerialConnector.java:83)> <INFO> : <Leitura finalizada>
<22/11/2012 01:43:18> <(SerialConnector.java:93)> <DEBUG> : <Valor do buffer: 123498765;CON

>
<22/11/2012 01:43:18> <(SerialConnector.java:94)> <INFO> : <<>>
<22/11/2012 01:43:18> <(Rotina.java:151)> <DEBUG> : <Valor do entradados: 123498765;CON

>
<22/11/2012 01:43:18> <(Rotina.java:176)> <DEBUG> : <Valor entradadados: 123498765;CON

>
<22/11/2012 01:43:47> <(Rotina.java:186)> <DEBUG> : <Testetoken é: 123498765;CON

>
<22/11/2012 01:43:47> <(Rotina.java:205)> <DEBUG> : <123498765>
<22/11/2012 01:43:47> <(Rotina.java:207)> <DEBUG> : <CON

>
<22/11/2012 01:44:04> <(Rotina.java:281)> <INFO> : <>
<22/11/2012 01:44:04> <(Rotina.java:282)> <INFO> : <Operação de Consulta>
<22/11/2012 01:44:04> <(Rotina.java:283)> <INFO> : <>
<22/11/2012 01:44:19> <(BancoOperacoes.java:171)> <INFO> : <>
<22/11/2012 01:44:19> <(BancoOperacoes.java:172)> <INFO> : <Setando variaveis para fazer a consulta no banco...>
<22/11/2012 01:44:19> <(BancoOperacoes.java:173)> <INFO> : <>
<22/11/2012 01:44:19> <(BancoOperacoes.java:174)> <INFO> : <Abrindo conexão...>
<22/11/2012 01:44:19> <(BancoOperacoes.java:175)> <INFO> : <>
<22/11/2012 01:44:20> <(BancoOperacoes.java:179)> <INFO> : <>
<22/11/2012 01:44:20> <(BancoOperacoes.java:180)> <INFO> : <populando dados do select...>
<22/11/2012 01:44:20> <(BancoOperacoes.java:181)> <INFO> : <>
<22/11/2012 01:44:20> <(BancoOperacoes.java:184)> <DEBUG> : <sql: SELECT PRECO , NOME FROM VW_CONSULTA_PRECO WHERE TAG = '123498765' >
<22/11/2012 01:44:20> <(BancoOperacoes.java:186)> <INFO> : <Executando select...>
<22/11/2012 01:44:20> <(BancoOperacoes.java:187)> <INFO> : <>
<22/11/2012 01:44:20> <(BancoOperacoes.java:212)> <DEBUG> : <campos: [PRECO, NOME]>
<22/11/2012 01:44:20> <(BancoOperacoes.java:213)> <DEBUG> : <dados: [R$2,50, COCA COLA ZERO 600ML]>
<22/11/2012 01:44:20> <(BancoOperacoes.java:226)> <INFO> : <>
<22/11/2012 01:44:20> <(BancoOperacoes.java:227)> <INFO> : <Select Executado com sucesso!>
<22/11/2012 01:44:20> <(BancoOperacoes.java:228)> <INFO> : <>
<22/11/2012 01:44:20> <(BancoOperacoes.java:229)> <INFO> : <Fechando conexão...>
<22/11/2012 01:44:20> <(BancoOperacoes.java:230)> <INFO> : <>
<22/11/2012 01:44:20> <(Rotina.java:290)> <INFO> : <rs.size: 2>
<22/11/2012 01:44:20> <(Rotina.java:294)> <INFO> : <>
<22/11/2012 01:44:20> <(Rotina.java:295)> <INFO> : <Resultado da Consulta no banco:>
<22/11/2012 01:44:20> <(Rotina.java:296)> <INFO> : <DADO1: R$2,50>
<22/11/2012 01:44:20> <(Rotina.java:297)> <INFO> : <DADO2: COCA COLA ZERO 600ML>
<22/11/2012 01:44:20> <(Rotina.java:298)> <INFO> : <>
<22/11/2012 01:44:20> <(SerialConnector.java:112)> <INFO> : <Escrita iniciada>
<22/11/2012 01:44:20> <(SerialConnector.java:115)> <INFO> : <Escrita habilitada>
<22/11/2012 01:44:20> <(SerialConnector.java:120)> <INFO> : <Id porta obtido>

<22/11/2012 01:44:20> <(SerialConnector.java:125)> <INFO> : <Porta aberta>
<22/11/2012 01:44:20> <(SerialComLeitura.java:225)> <INFO> : <FLUXO OK!>
<22/11/2012 01:44:20> <(SerialComLeitura.java:235)> <INFO> : <Enviando um byte para COM1>
<22/11/2012 01:44:20> <(SerialComLeitura.java:237)> <DEBUG> : <Enviando : R$2,50>
<22/11/2012 01:44:20> <(SerialConnector.java:130)> <INFO> : <Dados enviados>
<22/11/2012 01:44:20> <(SerialConnector.java:156)> <INFO> : <Porta fechada>
<22/11/2012 01:44:20> <(SerialConnector.java:166)> <INFO> : <Escrita finalizada>
<22/11/2012 01:44:20> <(Rotina.java:306)> <INFO> : <Dados enviados>