

**UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ  
CAMPUS PATO BRANCO  
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE  
SISTEMAS**

**JEAN CARLO CANTÚ**

**SISTEMA PARA CONTROLE DE PONTO DE FUNCIONÁRIOS**

**TRABALHO DE CONCLUSÃO DE CURSO**

**PATO BRANCO  
2011**

**JEAN CARLO CANTÚ**

## **SISTEMA PARA CONTROLE DE PONTO DE FUNCIONÁRIOS**

Trabalho de Conclusão de Curso de graduação, apresentado à disciplina de Trabalho de Diplomação, do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, da Universidade Tecnológica Federal do Paraná, Campus Pato Branco, como requisito parcial para obtenção do título de Tecnólogo.

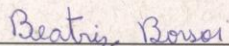
Orientadora: Profa. Beatriz Terezinha Borsoi

**PATO BRANCO  
2011**

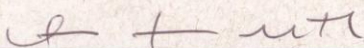
ATA Nº: 184

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO JEAN CARLO CANTÚ.

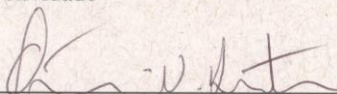
Às 11:05 hrs do dia 6 de julho de 2011, Bloco S da UTFPR, Campus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Omero Francisco Bertol (Convidado) e Géri Natalino Dutra (Convidado), para avaliar o Trabalho de Diplomação do aluno Jean Carlo Cantú, matrícula 846201, sob o título **Sistema para Controle de Ponto de Funcionários**; como requisito final para a conclusão da disciplina Trabalho de Diplomação do Curso Superior de Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas, Coordenação de Informática. Após a apresentação o candidato foi entrevistado pela banca examinadora, e a palavra foi aberta ao público. Em seguida, a banca reuniu-se para deliberar considerando o trabalho **APROVADO**. Às 11:55 hrs foi encerrada a sessão.



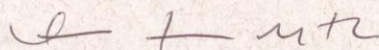
Prof. Beatriz Terezinha Borsoi, Dr.  
Orientadora



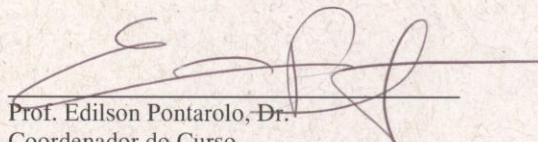
Prof. Omero Francisco Bertol, M.Sc.  
Convidado



Prof. Géri Natalino Dutra, M.Sc.  
Convidado



Prof. Omero Francisco Bertol, M.Sc.  
Coordenador do Trabalho de Diplomação



Prof. Edilson Pontarolo, Dr.  
Coordenador do Curso

## RESUMO

CANTÚ. Jean Carlo. Sistema para controle de ponto de funcionários. 2011. 60 f. Monografia de Trabalho de Conclusão de Curso. Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. Universidade Tecnológica Federal do Paraná. Pato Branco, 2011.

O controle eletrônico de ponto passou a ser uma exigência governamental para empresas que possuem funcionários a partir de uma determinada quantidade. Esses equipamentos registram o ponto e emitem comprovante de registro, mas é importante que esses registros possam ser automaticamente exportados para o setor de contabilidade, financeiro ou de recursos humanos da empresa. Assim, esse trabalho reporta o desenvolvimento de um sistema para o controle de ponto de funcionários. O sistema permite exportar os dados registrados pelo equipamento e exportá-los para um aplicativo gerencial da empresa. Para o desenvolvimento foi utilizada a linguagem Delphi com o banco de dados Firebird.

**Palavras-chave:** Controle de ponto. Linguagem Delphi. Banco de dados Firebird. Orientação a objetos.

## **AGRADECIMENTOS**

Agradeço aos meus familiares, que sem eles eu não teria nada, nem mesmo condições de fazer essa faculdade, sempre me dando forças para continuar e lutar pelos meus objetivos.

Aos meus amigos, que na hora que precisei de alguma ajuda em relação ao meu trabalho, sempre estavam ali prontos a me ajudar.

E aos meus professores, por me ensinarem e ajudarem a escolher uma profissão para o meu futuro.

## LISTA DE FIGURAS

Figura 1 – Exemplo de classe .....	15
Figura 2 – Diagramas da UML versão 2.....	16
Figura 3 – Exemplo de diagrama de caso de uso .....	17
Figura 4 – Exemplo de diagrama de classes.....	18
Figura 5 – Exemplo de diagrama de atividades .....	19
Figura 6 – Exemplo de diagrama de sequência .....	20
Figura 7 – Tela principal do Jude Community .....	22
Figura 8 – Tela principal do Borland Delphi 7.....	23
Figura 9 – Tela principal do IBExpert .....	25
Figura 10 – Tela de registro IBExpert.....	25
Figura 11 – Tela principal do Report Builder .....	27
Figura 12 – Diagrama de caso de uso nível administrador .....	30
Figura 13 – Diagrama de entidades e relacionamentos do sistema.....	31
Figura 14 – Diagrama de atividade inclusão de usuário.....	32
Figura 15 – Diagrama de atividade alteração de usuário .....	33
Figura 16 – Diagrama de atividade exclusão de usuário.....	34
Figura 17 – Diagrama de atividade inclusão da empresa .....	35
Figura 18 – Diagrama de atividade alteração de empresa.....	35
Figura 19 – Diagrama de sequência inclusão de usuário.....	36
Figura 20 – Diagrama de sequência alteração de usuário .....	37
Figura 21 – Diagrama de sequência exclusão de usuário.....	38
Figura 22 – Diagrama de sequência inclusão de empresa .....	39
Figura 23 – Diagrama de sequência alteração de empresa.....	40
Figura 24 – Tela de <i>login</i> .....	42
Figura 25 – Tela de seleção do estabelecimento .....	43
Figura 26 – Tela principal do sistema.....	43
Figura 27 – Manutenção de estabelecimentos.....	44
Figura 28 – Tela de alteração do cadastro de estabelecimento .....	45
Figura 29 – Tela de cadastro de eventos .....	45
Figura 30 – Tela de cadastro de eventos do sistema do ponto .....	46
Figura 31 – Tela de cadastro de leiaute de importação .....	46
Figura 32 – Itens do menu cadastros específicos .....	47
Figura 33 – Manutenção de empregados - aba principal .....	47
Figura 34 – Manutenção de empregados - aba endereço.....	48
Figura 35 – Manutenção de empregados - aba informações adicionais .....	48
Figura 36 – Tela de movimento.....	49
Figura 37 – Importação de ponto .....	49
Figura 38 – Tela de dados gerados (importação do ponto).....	50
Figura 39 – Relatório de eventos do ponto .....	50
Figura 40 – Modelo de relatório padrão .....	53
Figura 41 – Tela menu padrão .....	56
Figura 42 – Criando um relatório.....	57

## LISTA DE TABELAS

Quadro 1 – Fluxo principal dos casos de uso.....	31
Quadro 2 – Interfaces de usuário .....	41
Quadro 3 – Funções do sistema .....	41
Quadro 4 – Restrições do sistema .....	41

## LISTAGENS DE CÓDIGO

Listagem 1 - Código da <i>unit</i> de cadastro padrão.....	51
Listagem 2 - Código da unit de processo padrão .....	52
Listagem 3 - Código para seleção de arquivo em um diretório .....	53
Listagem 4 - Código da função LeLinha .....	54
Listagem 5 - Código da importação de ponto.....	55
Listagem 6 - Parte do código da função leComRegistro .....	55
Listagem 7 - Parte do código da função GravaDados.....	56
Listagem 8 - Código para chamada de relatório de ponto.....	56
Listagem 9 - Código de chamada de relatório de ponto .....	58



## LISTA DE SIGLAS

CASE	<i>Computer Aided Software Engineering</i>
IP	<i>Internet Protocol</i>
IPX/SPX	<i>Internetwork Packet Exchange/Sequenced Packet Exchange</i>
JUDE	<i>Java and UML Developer Environment</i>
OOA	<i>Object Oriented Analysis</i>
OOD	<i>Object Oriented Design</i>
OOP	<i>Object Oriented Programming</i>
SGBRD	<i>Sistema Gerenciador de Banco de Dados Relacional</i>
SQL	<i>Structured Query Language</i>
TCP/IP	<i>Transmission Control Protocol/Internet Protocol</i>
UML	<i>Unified Modeling Language</i>

## SUMÁRIO

1 INTRODUÇÃO .....	10
1.1 CONSIDERAÇÕES INICIAIS .....	10
1.2 OBJETIVOS .....	11
1.2.1 Objetivo Geral .....	11
1.2.2 Objetivos Específicos .....	11
1.3 JUSTIFICATIVA .....	11
1.4 ORGANIZAÇÃO DO TEXTO.....	12
2. FUNDAMENTAÇÃO TEÓRICA.....	13
2.1 ORIENTAÇÃO A OBJETOS.....	13
2.2 LINGUAGEM DE MODELAGEM UNIFICADA .....	15
3 MATERIAIS E MÉTODO .....	21
3.1 TECNOLOGIAS E FERRAMENTAS UTILIZADAS .....	21
3.1.1 JUDE Community .....	21
3.1.2 Linguagem Delphi .....	23
3.1.3 Banco de Dados Firebird.....	23
3.1.4 IBExpert.....	24
3.1.5 Report Builder .....	26
3.2 ATIVIDADES PARA MODELAGEM E IMPLEMENTAÇÃO DO SISTEMA.....	28
4 DESENVOLVIMENTO DO PROJETO .....	29
4.1 APRESENTAÇÃO DO SISTEMA.....	29
4.2 MODELAGEM DO SISTEMA.....	29
4.3 DESCRIÇÃO DO SISTEMA .....	42
4.4 IMPLEMENTAÇÃO DO SISTEMA .....	51
5 CONCLUSÃO.....	59
REFERÊNCIAS.....	60

## 1 INTRODUÇÃO

Este capítulo apresenta as considerações iniciais com uma visão geral do trabalho, os objetivos e a justificativa do trabalho, bem como a organização do texto.

### 1.1 CONSIDERAÇÕES INICIAIS

Os sistemas informatizados têm se tornado mais complexos no sentido das funcionalidades de negócio que eles provêm e do tipo de negócio que eles atendem. Eles têm fornecido informações para apoio à tomada de decisões, otimização de processos, redução de despesas e maximização de lucros.

Para que os requisitos definidos para o sistema pelos usuários sejam atendidos pode ser necessário um período longo de trabalho por uma equipe de profissionais, incluindo analistas, desenvolvedores e testadores, dentre outros. Contudo, é comum que equipes de desenvolvimento estejam sob pressão para desenvolver sistemas dentro de um prazo curto, do valor estipulado e com os requisitos de qualidade requeridos pelo usuário.

Uma das formas que podem ser indicadas para redução do prazo é o planejamento adequado do projeto e a realização de uma modelagem completa e efetiva do sistema a ser implementado. Modelagem efetiva se refere à que representa com clareza e completeza os requisitos do sistema. O uso de orientação a objetos permite a elaboração de uma modelagem que representa o sistema como um conjunto de objetos que interagem por meio da troca de mensagens (RUMBAUGH, 1997). Essa forma de entendimento do sistema pode auxiliar a representar o sistema de maneira mais próxima às necessidades e interesses do usuário e a portar esses requisitos para a implementação em código.

Contudo, o uso de orientação a objetos por si só não garante uma modelagem ou mesmo implementação adequadas. É necessário que o processo de análise e o registro dessas informações sejam feitos adequadamente. A UML (*Unified Modeling Language*) oferece uma forma de sistematização para modelar sistemas utilizando orientação a objetos. Os diversos tipos de diagramas que a mesma possui permite a representação do sistema sob pontos de vista e interesses distintos. E, assim, uma representação mais completa do sistema pode ser obtida.

Considerando a importância da orientação a objetos e do uso de uma

linguagem para representar os seus modelos, este trabalho reporta o desenvolvimento de um sistema que gerencia os registros de entrada e saída dos funcionários de uma empresa. Esse gerenciamento é realizado por meio da validação do arquivo ponto. A modelagem e a implementação do sistema têm como base a orientação a objetos.

## **1.2 OBJETIVOS**

O objetivo geral se refere ao resultado principal esperado com a realização deste trabalho. Os objetivos específicos complementam o objetivo geral.

### **1.2.1 Objetivo Geral**

Implementar um sistema para controlar os horários de entrada e de saída de funcionários de uma empresa, o denominado registro de ponto.

### **1.2.2 Objetivos Específicos**

Para que a implementação do sistema possa ser realizada é necessário:

- Representar as funções básicas do sistema por meio de diagramas da UML, modelando o problema e a solução proposta.
- Implementar relatórios que permitam controlar o registro dos horários de entrada e saída, identificando faltas e horas extras.
- Possibilitar o cadastro do leiaute do arquivo de importação de ponto.
- Permitir a importação dos registros de entrada e saída de funcionários do equipamento que faz o registro de ponto para o programa de controle da folha de pagamento da empresa.

## **1.3 JUSTIFICATIVA**

O controle dos horários de trabalho dos funcionários das empresas costumava ser feito manualmente por meio de um Livro de Ponto. Com a chegada de sistemas computacionais em muitas empresas o ponto tem sido marcado por

meio do uso de tecnologias como o reconhecimento de digital (biometria).

Contudo, de acordo com a legislação brasileira em vigor, uma empresa que possui mais de 10 funcionários está obrigada a registrar a entrada e saída dos seus funcionários por meio do ponto eletrônico, gerando um arquivo com as movimentações. Esse arquivo precisa ser posteriormente importado para o sistema de folha de pagamento.

Verificou-se assim a necessidade de desenvolver um sistema que realizasse a importação dos dados do equipamento de registro de ponto e os exportasse para a folha de pagamento. A solução apresentada como resultado deste trabalho foi desenvolvida para uma empresa específica. Contudo, essa solução pode ser utilizada por outras empresas. Isso porque o leiaute de importação do arquivo pode ser definido e configurado de acordo com o registro gerado pelo equipamento. Ressalta-se, porém que pode ser necessário adaptar o sistema para interação com o sistema de controle de folha de pagamento utilizado pela empresa.

#### **1.4 ORGANIZAÇÃO DO TEXTO**

Este texto está organizado em capítulos, dos quais este é o primeiro e apresenta a ideia do sistema, incluindo os objetivos e a justificativa.

No Capítulo 2 está o referencial teórico sobre orientação a objetos e a linguagem de modelagem UML.

No Capítulo 3 está o método, que é a sequência geral de passos para o ciclo de vida do sistema, e os materiais utilizados. Os materiais se referem ao que é necessário para modelar e implementar o sistema, incluindo as tecnologias, as ferramentas e os ambientes de desenvolvimento utilizados.

O Capítulo 4 contém a modelagem do sistema e exemplos da sua implementação, incluindo telas e códigos.

No Capítulo 5 está a conclusão com as considerações finais.

## 2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica deste trabalho está centrada em orientação a objetos porque esses conceitos foram utilizados tanto na documentação da análise e projeto quanto na sua implementação do sistema. Este capítulo também abrange a UML que define uma forma de especificar e documentar a modelagem orientada a objetos.

### 2.1 ORIENTAÇÃO A OBJETOS

O paradigma da orientação a objetos é baseado na construção de sistemas a partir de componentes reutilizáveis e permite que objetos do mundo real possam ser mapeados em objetos como tratados por linguagens de modelagem e de programação.

A fundamentação do modelo de objetos divide-se em OOA (*Object Oriented Analysis* - Análise Orientada a Objetos), OOD (*Object Oriented Design* - Projeto Orientado a Objetos) e OOP (*Object Oriented Programming* - Programação Orientada a Objetos) (RICARTE, 2011).

A análise no paradigma de orientação a objetos determina o que o sistema deve fazer. É uma técnica de análise que examina requisitos pelas perspectivas de classes e de objetos presentes no domínio do problema. O projeto orientado a objetos define um método de projeto para criar e descrever uma estrutura geral do sistema. E a programação trata da implementação do sistema. A programação orientada a objetos possui uma organização em termos de coleção de objetos incorporando estrutura e comportamento próprios (SIEDLER, 2011).

Dentre os conceitos básicos ou fundamentais relacionados à orientação a objetos estão: objeto, classe, encapsulamento, polimorfismo e herança.

Martin (1995, p. 18) define um objeto como sendo “qualquer coisa, real ou abstrata, a respeito da qual armazenamos dados e os métodos que os manipulam”. Os objetos podem ser representados como entidades concretas (um arquivo de texto) ou entidades conceituais (uma estratégia de jogo).

Um objeto deve conter uma identidade, atributos e métodos. A identidade é um nome ou uma identificação única e que individualiza o objeto de maneira

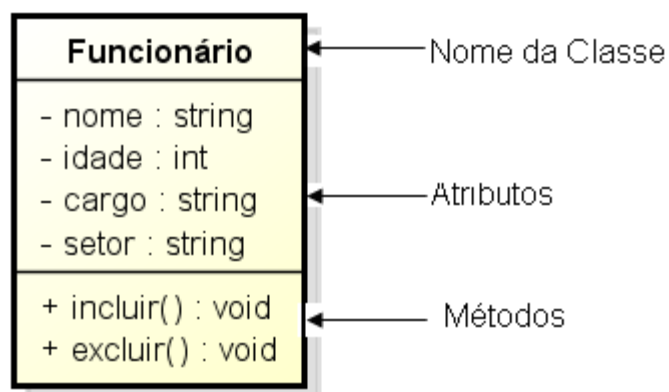
exclusiva dentre os demais objetos da mesma classe. Os atributos representam o estado do objeto no mundo real. Os atributos armazenam as características dos objetos e indicam as possíveis informações armazenadas em um objeto. Os métodos definem o comportamento dos objetos por meio de operações que podem ser executadas sobre os seus dados. Esses dados são os valores dos atributos.

A maneira como os dados de um objeto são manipulados e como se comportam é denominado operação. Uma operação quando implementada é chamada método. Uma operação é uma ação de envio de mensagens que é executada por um objeto. Os métodos de um tipo de objeto referenciam somente as estruturas de dados desse tipo de objeto.

Para fazer com que um objeto execute uma operação é preciso enviar uma solicitação ao mesmo. Essa solicitação é uma mensagem para executar a operação indicada sobre determinado objeto e retornar o resultado (MARTIN, 1995). As mensagens são constituídas pelo nome do objeto, o nome da operação e, eventualmente, um grupo de parâmetros. Os parâmetros são os dados enviados para a operação.

Uma classe abrange um conjunto de objetos com características similares. Uma classe determina o comportamento dos objetos por meio dos métodos e os seus possíveis estados por meio dos atributos. Para Martin (1995, p. 26) “uma classe é uma implementação de um tipo de objeto. Ela especifica uma estrutura de dados e os métodos operacionais permissíveis que se aplicam a cada um de seus objetos”.

A Figura 1 apresenta um exemplo de classe denominada Funcionário. Essa classe contém quatro atributos para armazenamento dos dados: nome, idade, cargo e setor; e os métodos, ou operações, que executam sobre os dados dessa classe que são: incluir e excluir. Nesse exemplo não estão indicados os valores que a operação recebe (parâmetros) e nem o seu retorno. Porém, ressalta-se que um método pode não receber parâmetros e/ou não retornar valor, isto é ser do tipo *void*, por exemplo.



**Figura 1 – Exemplo de classe**

Encapsulamento é o processo de empacotar dados e métodos ao mesmo tempo. Esse mecanismo é utilizado para impedir o acesso aos atributos do objeto, disponibilizando externamente apenas os métodos que alteram tais atributos. Os usuários entendem as operações do objeto que podem ser solicitadas, porém não conhecem os detalhes da execução da operação. Toda a informação encapsulada pode ser modificada sem que os usuários da classe sejam afetados.

Polimorfismo significa que uma mesma operação pode ter um comportamento diferente para classes diferentes. Assim, um método pode apresentar várias formas, de acordo com seu contexto. O polimorfismo permite que a mudança de uma interface seja efetivamente separada da implementação que a representa.

Herança é o processo que permite o compartilhamento de atributos e de operações entre classes baseada em uma hierarquia. Martin (1995) explica que uma subclasse herda as propriedades da sua classe mãe e uma subclasse herda as propriedades das subclasses e assim sucessivamente. As propriedades da classe base não precisam ser repetidas em cada classe derivada. Essa hierarquia de classes e suas propriedades podem reduzir a repetição de código em um *software*.

## 2.2 LINGUAGEM DE MODELAGEM UNIFICADA

A Linguagem de Modelagem Unificada, a UML, é uma linguagem para o desenvolvimento da estrutura de projetos de *software*, no sentido de documentar os resultados da análise e do projeto. Esses resultados definem o sistema que será implementado por uma linguagem de programação visando atender os interesses dos usuários desse sistema. A UML possui diversas funcionalidades que



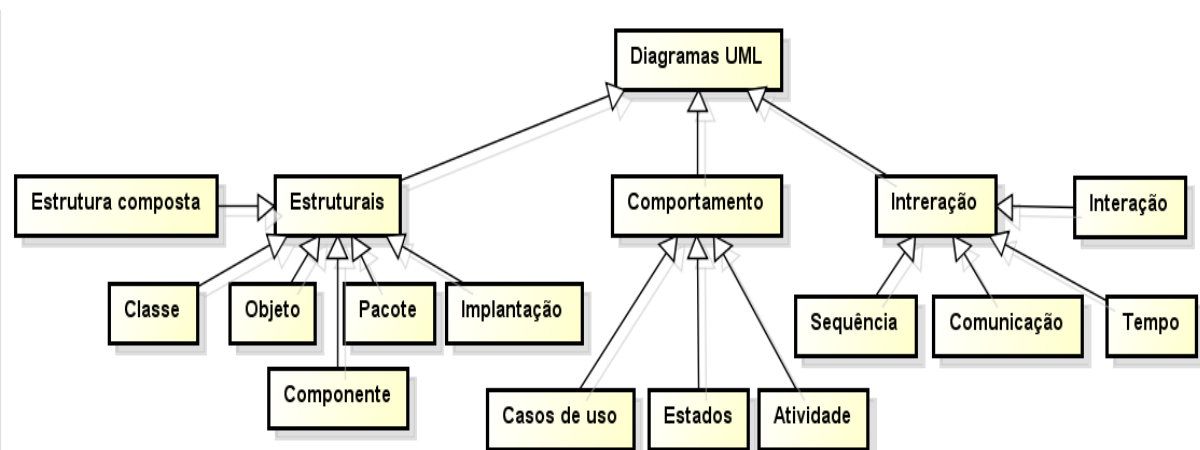
possibilitam visualização, especificação, construção e documentação de artefatos, que podem ser elementos relacionados à especificação de um *software*.

Para Booch, Rumbaugh e Jacobson (2000) a UML é adequada para a modelagem de sistemas, que podem abranger sistemas de informação corporativos, aplicações baseadas em *web* e até sistemas complexos de tempo real que incorporam *software* e *hardware*. A UML permite representar as diversas visões que podem ser necessárias para o desenvolvimento e implantação de sistemas.

Com a utilização da modelagem de *software*, os modelos podem ser usados na identificação das características e funcionalidades que o *software* deverá atender. Além disso, a UML é composta por elementos que representam as diferentes partes de um sistema de *software*. Tais elementos são usados na criação de diagramas para representar determinadas partes ou pontos de vista do sistema.

As diversas representações de um sistema possíveis com os diagramas da UML facilitam a compreensão desse sistema como um todo. Essas representações facilitam a visualização e o entendimento. Isso porque interesses distintos são representados por diagramas distintos. Essas representações são realizadas por meio de diagramas que são modelos gráficos compostos por um conjunto de elementos conectados entre si.

A Figura 2 representa a divisão dos diagramas UML na sua versão 2.0. Essa figura foi composta a partir das informações textuais constantes em UML 2.0 (2011). A UML 2.0 define treze tipos de diagramas, agrupados em três categorias. São seis tipos de diagramas que representam a estrutura estática da aplicação, três tipos que representam comportamento e quatro tipos que representam diferentes aspectos de interação.



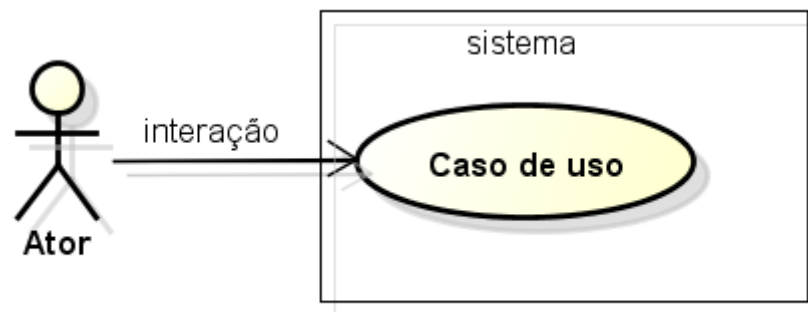
**Figura 2 – Diagramas da UML versão 2**

Fonte: composto com base em informações constantes em UML 2.0 (2011)

A seguir alguns dos diagramas da Figura 2 são descritos. Foram escolhidos os diagramas de casos de uso, classes e atividades por serem os utilizados na representação da modelagem do sistema desenvolvido como resultado deste trabalho.

Diagramas de casos de uso fornecem um modo de descrever a visão externa do sistema e suas interações com o mundo exterior. Assim, podem representar uma visão de alto nível de funcionalidades intencionais mediante requisições feitas pelo usuário. Os diagramas de caso de uso apresentam uma visão externa sobre como esses elementos podem ser utilizados no contexto do sistema sendo representado (FERREIRA, 2009).

A Figura 3 representa um exemplo genérico de diagrama de caso de uso em que um ator interage com alguma função do sistema. A função é representada pelo caso de uso. A interação pode ser do ator para com o caso de uso ou do caso de uso para com o ator, isso no sentido de quem inicia a interação.



**Figura 3 – Exemplo de diagrama de caso de uso**

Um diagrama de classes é uma estrutura lógica estática que mostra uma coleção de elementos declarativos de modelo, como classes, tipos e seus respectivos conteúdos e relações. Na Figura 4 está um exemplo genérico de classes expondo relacionamentos associação, herança, composição e agregação.

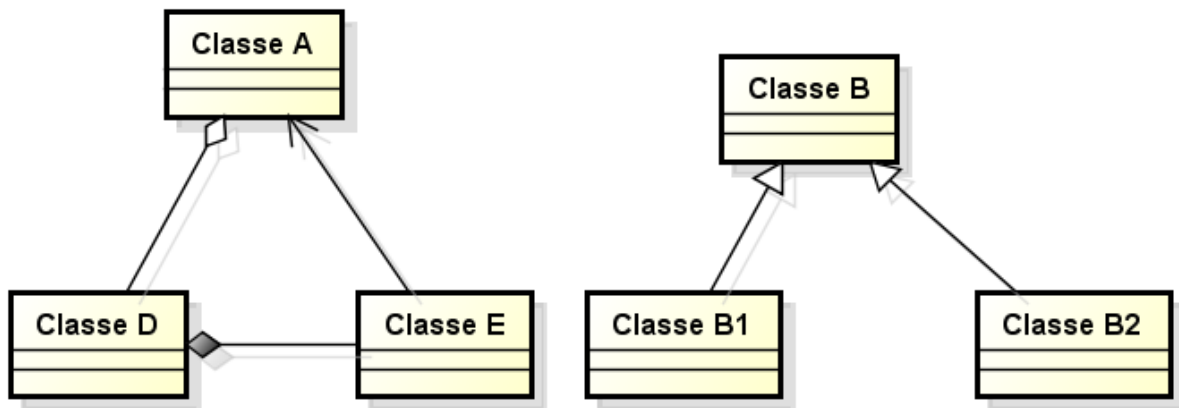
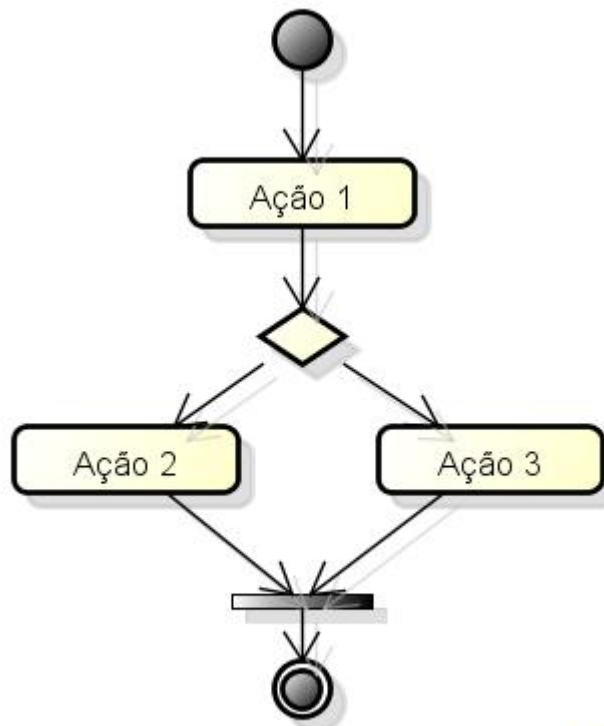


Figura 4 – Exemplo de diagrama de classes

Um diagrama de atividades é essencialmente um gráfico de fluxo, mostrando o fluxo de controle de uma atividade para outra. Esse tipo de diagrama é utilizado para descrever um comportamento paralelo ou mostrar como interagem comportamentos em vários casos de uso. Podem permanecer isolados para visualizar, especificar, construir e documentar a dinâmica de uma variedade de objetos, ou poderão ser utilizados na modelagem do fluxo de controle de uma operação.

A Figura 5 apresenta a ideia geral de um diagrama de atividades. Esse diagrama tem indicação de início e de fim e as ações ou atividades são representadas por meio de fluxos que as interconectam. A conexão das atividades pode ser realizada utilizando decisão entre caminhos possíveis e a junção entre ações, dentre outros.



**Figura 5 – Exemplo de diagrama de atividades**

O diagrama de sequência expõe o aspecto do modelo que destaca o comportamento dos objetos em um sistema, incluindo suas operações, interações, colaborações, histórias de estado e sequência temporal de mensagem e representação explícita de atividades de operações (FURLAN, 1998). O diagrama de sequência serve para enfatizar a ordenação temporal das mensagens.

A Figura 6 mostra um exemplo genérico de diagrama de sequência que contém a ordem temporal de execução dos processos. Um ator ou método faz uma solicitação a outro ator ou método que devolve a resposta para a continuação do processo. Um método pode ser representado como componente, classe, formulário, outro sistema ou mesmo um equipamento, como uma impressora, por exemplo.

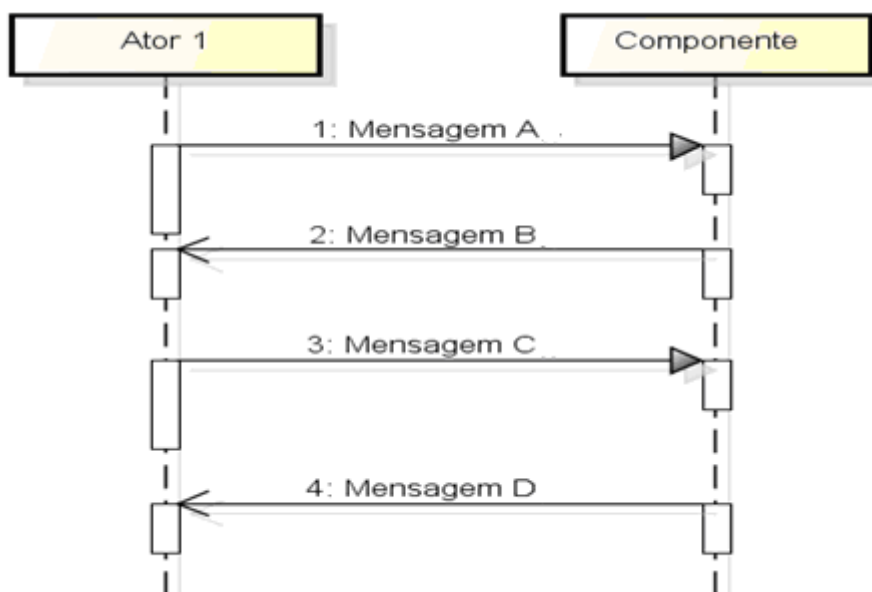


Figura 6 – Exemplo de diagrama de sequência

### 3 MATERIAIS E MÉTODO

Os materiais se referem às tecnologias e ferramentas utilizadas para modelar e implementar o sistema. O método está relacionado as principais atividades desenvolvidas para a realização deste trabalho.

#### 3.1 TECNOLOGIAS E FERRAMENTAS UTILIZADAS

As ferramentas e as tecnologias utilizadas no desenvolvimento deste trabalho foram o JUDE Community para a construção de diagramas, a linguagem Delphi como linguagem de programação, o banco de dados Firebird com o IBExpert como administrador do banco de dados e o Report Builder para a construção dos relatórios.

A escolha da ferramenta JUDE (*Java and UML Developer Environment*) foi devido a sua facilidade de interação com o usuário e por ser gratuita. Com essa ferramenta foi possível desenvolver todos os diagramas necessários para a modelagem do sistema que é resultado deste trabalho.

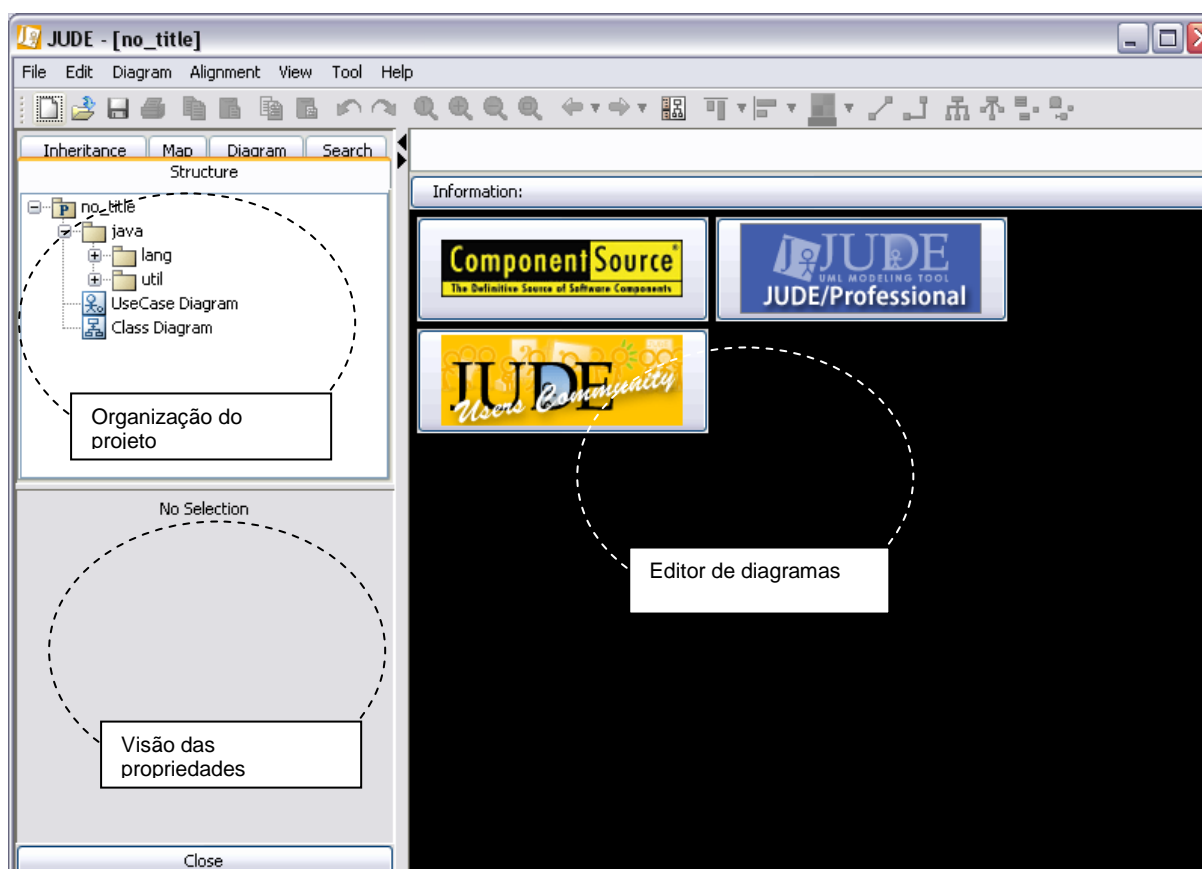
O banco de dados Firebird foi utilizado por ser um banco de dados gratuito e ter uso amplo. Isso é verificado pela quantidade de exemplos de uso, além de fóruns de listas de discussões encontrados na Internet. O projeto Firebird é aberto à comunidade para participação no desenvolvimento do código desse banco de dados, correção de erros, documentação, desenvolvimento de ferramentas, dentre outros.

##### 3.1.1 JUDE Community

A ferramenta CASE (*Computer Aided Software Engineering*) utilizada no desenvolvimento do projeto tornou possível uma melhor visualização do sistema como um todo. Na Figura 7 é apresentada a tela principal da ferramenta JUDE Community.

JUDE é uma IDE (*Integrated Development Environment*) para modelagem de dados criada com Java e de uso fácil e intuitivo (CAMPOS, 2008). JUDE possibilita, ainda, a Java Reverse que é a importação de código Java para criar um modelo e

Java Forward que é a geração de código fonte Java a partir de um modelo.



**Figura 7 – Tela principal do Jude Community**

A ferramenta JUDE está organizada em três partes básicas (representadas pelas áreas circuladas na Figura 7):

a) Organização do projeto - é uma área que possui várias abas com visões diferentes do projeto, são elas: *Structure* (árvore de estrutura do projeto), *Inheritance* (exibe as heranças identificadas), *Map* (exibe todo o editor de diagrama), *Diagram* (mostra a lista de diagramas do projeto), *Search* (para localização de modelos e substituição de nomes).

b) Visão das propriedades - é a área na qual podem ser alteradas as propriedades dos elementos dos diagramas. As propriedades do elemento selecionado, que está na área de edição, são apresentadas nesta área.

c) Editor de diagrama - é a área em que os diagramas são compostos e exibidos. Os elementos para compor os diagramas estão na barra superior a essa área. O tipo de diagrama escolhido determina os elementos disponíveis para uso. O tipo de diagrama é escolhido por meio de opções do menu *Diagram*.

### 3.1.2 Linguagem Delphi

O ambiente de desenvolvimento visual e orientado a objetos Borland® Delphi 7 é um ambiente integrado para desenvolvimento de *software*, produzido e mantido pela CodeGear<sup>1</sup>, antigamente conhecida como *Borland Software Corporation*. A linguagem utilizada pelo Delphi é o *Object Pascal*, derivada da linguagem de programação estruturada Pascal (CANTÙ, 2003).

Uma grande vantagem do desenvolvimento em Delphi, é o seu suporte nativo a vários bancos de dados, principalmente utilizando os componentes *dbExpress*. Além de possibilitar o desenvolvimento de *software* estruturado em multicamadas. A Figura 8 apresenta a janela principal da IDE Delphi.

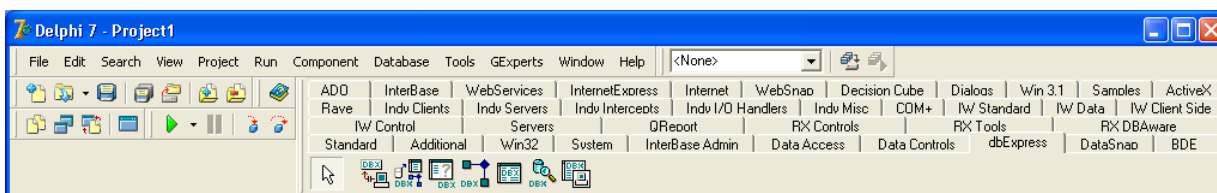


Figura 8 – Tela principal do Borland Delphi 7

### 3.1.3 Banco de Dados Firebird

Para o desenvolvimento do *software* proposto foi utilizado o banco de dados Firebird, aproveitando uma das principais características de um sistema multicamadas, com acesso a banco de dados por meio dos componentes *dbExpress*, que é a portabilidade. Esse banco de dados permite acesso de várias formas diferentes e também acessar vários bancos de dados sem alteração na codificação do sistema. A única preocupação quanto a isso se deve a conversão de *procedures*, *views*, *triggers* e tabelas de um banco de dados para outro.

O Firebird é um sistema gerenciador de banco de dados relacional (SGBDR), SQL (*Structured Query Language*) Cliente/Servidor que pode ser executado em uma diversidade de plataformas de sistemas operacionais servidores e clientes (CANTU, 2010). O Firebird é desenvolvido como um projeto *Open Source* (ou código aberto)<sup>2</sup>, que descende do código aberto do *Interbase* na versão 6.0 da Borland®.

<sup>1</sup> <http://www.codegear.com/>, visitado em 29/03/2011.

<sup>2</sup> <http://www.firebirdsql.org/>, visitado em 29/03/2011.



O banco de dados Firebird possui somente um arquivo de banco. Nesse arquivo estarão todos os elementos relacionados ao banco de dados, incluindo tabelas, procedimentos armazenados e configurações. Esse banco fornece suporte a campos *blob* que podem armazenar qualquer tipo de informações nele e, ainda, suporte, aos diversos protocolos como: local, TCP/IP (*Transmission Control Protocol/Internet Protocol*), NetBeui, IPX/SPX (*Internetwork Packet Exchange/Sequenced Packet Exchange*) (Novell).

Os códigos fonte do Firebird são mantidos por uma comunidade de desenvolvedores estruturada a partir do *SourceForge*. Na página dessa comunidade (<http://www.sourceforge.net>) é possível encontrar todo o código fonte do desse banco e suporte dos programadores.

#### **3.1.4 IBExpert**

O IBExpert (IBEXPERT, 2011) é uma ferramenta para administração de bancos de dados Interbase e Firebird. Essa ferramenta permite criar e gerenciar usuários, tabelas, *triggers*, *procedures*, *views*, etc.

Para utilizar o IBExpert é necessário registrar o banco de dados. Após esse registro é possível realizar as operações necessárias no banco de dados. O IBExpert possui uma versão gratuita para testes. Com o IBExpert é possível analisar dados, copiar objetos de bancos de dados, utilizar ferramentas de SQL, comparar bancos de dados ou tabelas de bancos de dados, bem como a opção de *Database designer*, para criar tabelas.

A Figura 9 apresenta a tela principal da ferramenta IBExpert. A área à direita é destinada à edição e os demais elementos estão na área à esquerda e na barra superior de ícones. E, ainda, nas opções dos menus.

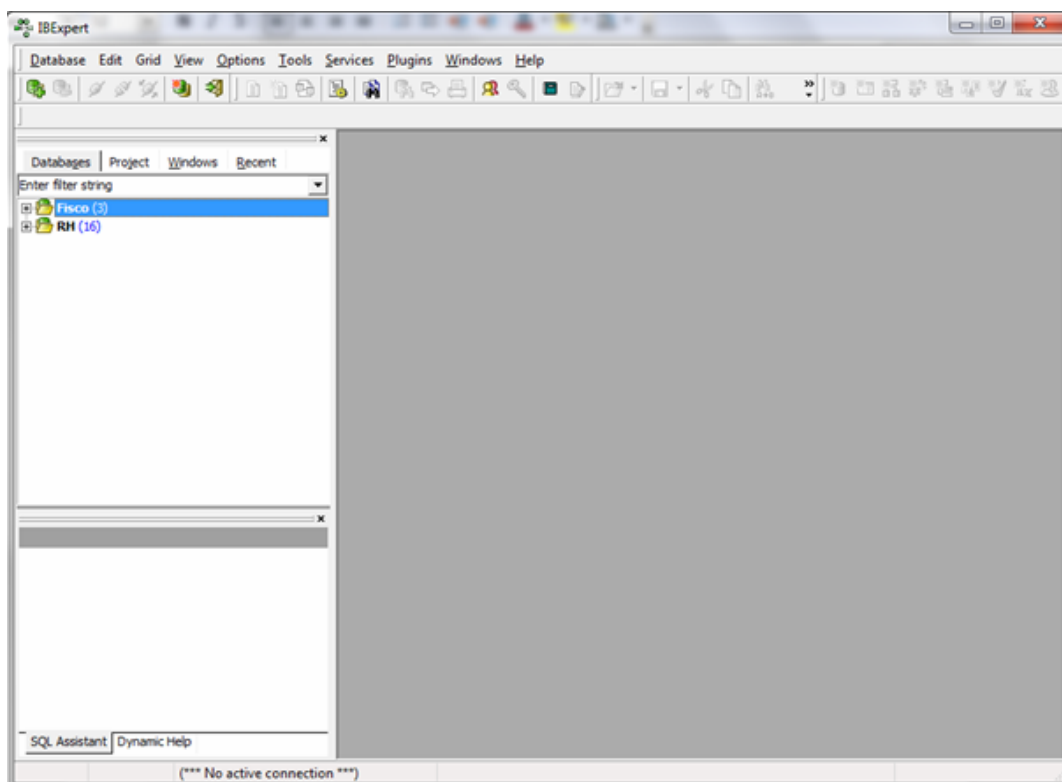


Figura 9 – Tela principal do IBEExpert

A criação de um banco de dados é feita pela opção *Create Database* do Menu *Database*. A Figura 10 apresenta a tela de criação de um novo banco de dados.

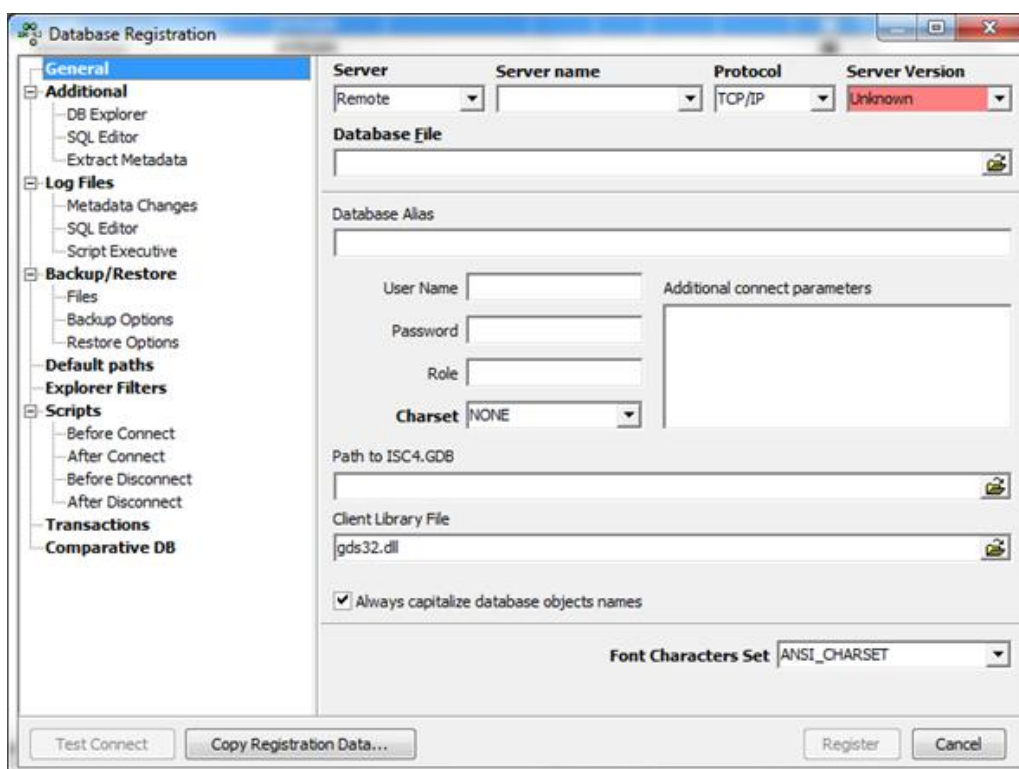


Figura 10 – Tela de registro IBEExpert

Os dados a serem indicados na criação de um banco de dados no IBEExpert,

de acordo com a Figura 10, são:

a) *Server* - determina o tipo de servidor, se remoto ou local. A opção remoto permite optar pela criação em outras máquinas de uma rede.

b) *Server Name* - se escolhida a opção remoto no campo anterior, é necessário informar o nome ou IP (*Internet Protocol*) do computador servidor da base de dados.

c) *Protocol* - indica o protocolo usado para efetuar a comunicação com o servidor. Selecionando a opção de servidor local, os campos *Server Name* e *Protocol* são desabilitados.

d) *Database* - nesse campo é informado o caminho da base de dados seguido do nome da base de dados acrescido de sua extensão.

e) *Client Library Name* - nesse campo é informada a biblioteca .dll (*Dymanic-Link Library*) do banco.

Os outros campos são Usuário/Senha (*SYSDBA/masterkey*) do Firebird, tamanho da página do arquivo, o *charset* (por exemplo, WIN1252), o dialeto SQL e a opção para registrar a base de dados após sua criação.

O banco de dados criado *precisa* ser registrado. Para isso é necessário informar a versão do banco de dados e o *alias* a ser referenciado do arquivo, o banco de dados no DBExplorer.

### 3.1.5 Report Builder

O ReportBuilder (Figura 11) é um ambiente de desenvolvimento que pode ser usado para construir relatórios, componentes do relatório e aplicações de informação. É uma ferramenta de desenvolvimento que pode ser utilizado na programação Delphi. A interface do Report Builder tem como base os leiautes do Microsoft Office, tais como o Excel e o PowerPoint. Ele fornece um ambiente rápido e uma plataforma orientada a objetos para resolver a solução de relatórios.

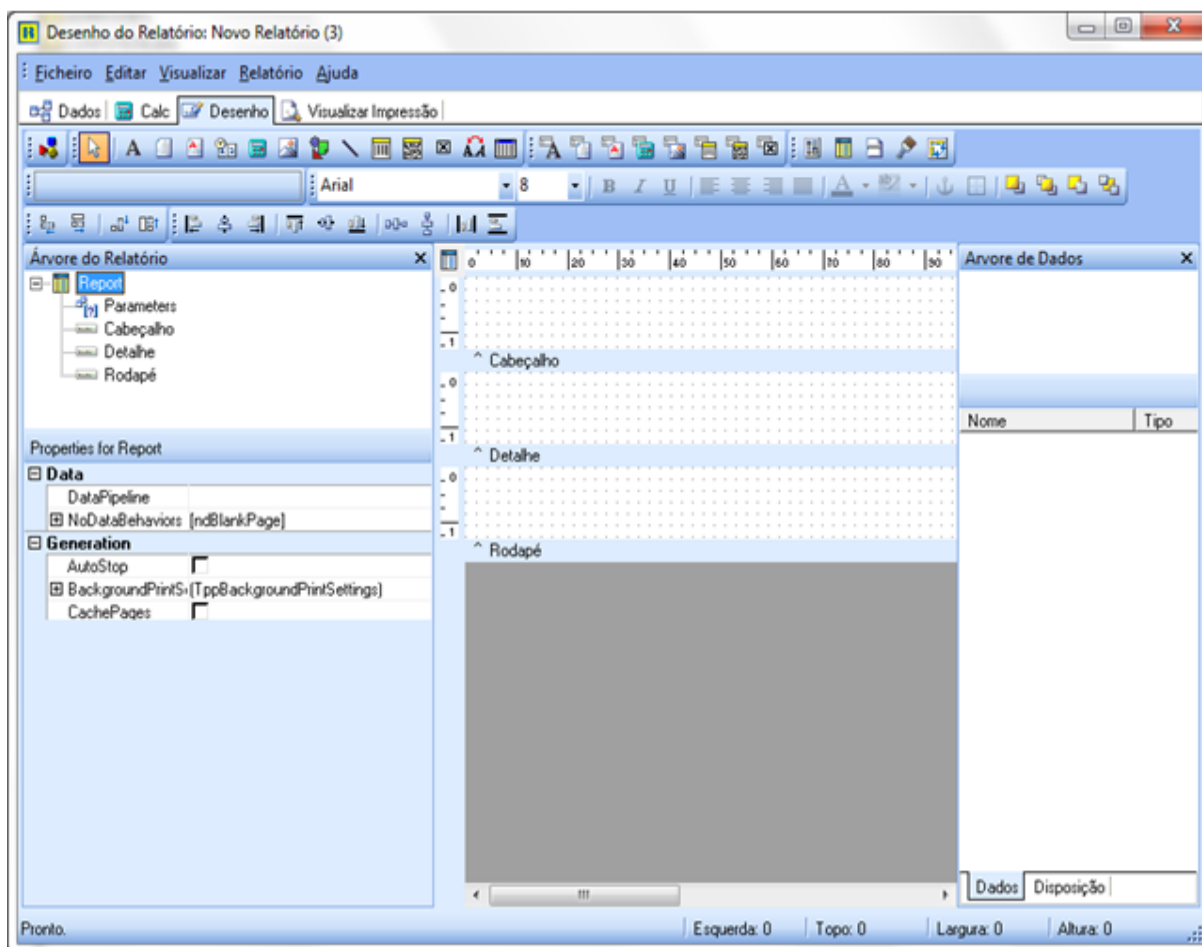


Figura 11 – Tela principal do Report Builder

O Report Builder prevê a recuperação de dados de uma tabela do banco de dados, arquivo de texto, objeto do Delphi, ou outra fonte de dados estruturada como registros e campos. A integração com a linguagem Delphi em tempo de projeto permite o uso da *inspector* do Delphi e o editor de código para implementar manipulador eventos do relatório.

No próprio Report Builder existem leiautes pré-definidos que podem ser utilizados nos relatórios desenvolvidos. A criação de relatórios pode ser feita por ferramentas de arrastar e soltar. É possível também fazer o controle sobre as propriedades da impressora, como: orientação, tamanho do papel, bandeja de papel duplex, margens, dentre outros. A ferramenta fornece suporte para exportação do relatório em arquivos externos como pdf, doc, txt.

### 3.2 ATIVIDADES PARA MODELAGEM E IMPLEMENTAÇÃO DO SISTEMA

As etapas para a modelagem e a implementação do sistema para controle de ponto de funcionários seguiram o modelo sequencial linear proposto por Pressman (2005). O uso desse modelo é justificado porque o sistema é simples e os requisitos do mesmo foram completamente definidos no início do processo. Isso foi possível pelo fato de que o sistema ter sido desenvolvido para a empresa na qual o autor deste relatório trabalha e pelo mesmo ter conhecimento das funcionalidades que o sistema deveria ter.

As etapas definidas foram:

a) Levantamento dos requisitos

Os requisitos foram definidos tendo como base a necessidade de um controle do registro de ponto utilizado pela própria empresa e outros sistemas desenvolvidos pela empresa que definem alguns padrões e especificações mínimas de um sistema. É exigência legal que as empresas que possuem registro ponto façam um controle do horário cumprido pelo funcionário. Essa exigência auxiliou a determinar as informações que o sistema deveria manipular e a integração com outros sistemas.

b) Análise e projeto

Os requisitos levantados foram modelados por meio de diagrama de casos de uso, classes, atividades e sequência. As tabelas e respectivos campos do banco de dados foram definidos a partir das classes identificadas como persistentes.

c) Implementação

Na implementação inicialmente foi definida a base de dados. A composição da base em Firebird foi realizada por meio da ferramenta IBExpert. A implementação do sistema foi realizada por meio da linguagem Delphi. Os relatórios foram elaborados utilizando o Report Builder.

d) Testes

Os testes foram realizados pelo autor deste trabalho visando identificar erros de codificação e testes para verificar se as funcionalidades do sistema foram devidamente implementadas. O sistema está em uso pela empresa. Na fase inicial de implementação foram realizados testes de usuários e de funcionalidades e as inconsistências identificadas já foram devidamente ajustadas.

## **4 DESENVOLVIMENTO DO PROJETO**

Este capítulo apresenta o resultado prático da realização deste trabalho.

### **4.1 APRESENTAÇÃO DO SISTEMA**

O sistema analisado, modelado e implementado tem como finalidade a importação do arquivo ponto de registro de entrada e saída dos funcionários de uma empresa.

Para que o sistema possa ser utilizado, inicialmente é preciso cadastrar todos os funcionários da empresa, os eventos e o leiaute de importação do ponto. A identificação será feita por meio de um código individual para cada funcionário no ponto. Os operadores ficarão responsáveis pelo cadastro e controle do sistema.

Cada funcionário terá informado no seu cadastro o seu respectivo código na máquina do ponto. Assim, ao ser importado o arquivo do equipamento poderá ser associado ao cadastro do funcionário. O sistema irá ler o arquivo do ponto e lançar em uma tabela o total de horas normais, horas extras e horas de faltas que o funcionário possui, conforme o arquivo importado.

Por meio dessa técnica de registro de ponto, pretende-se melhorar o controle dos horários dos funcionários da empresa, verificando o total de horas extras feitas pelo funcionário, ou o total de horas de faltas do mesmo. Esses valores são visualizados por meio de relatórios gerados pelo sistema.

### **4.2 MODELAGEM DO SISTEMA**

O diagrama de caso de uso, mostrado na Figura 12, têm por finalidade descrever e definir os requisitos funcionais do sistema.

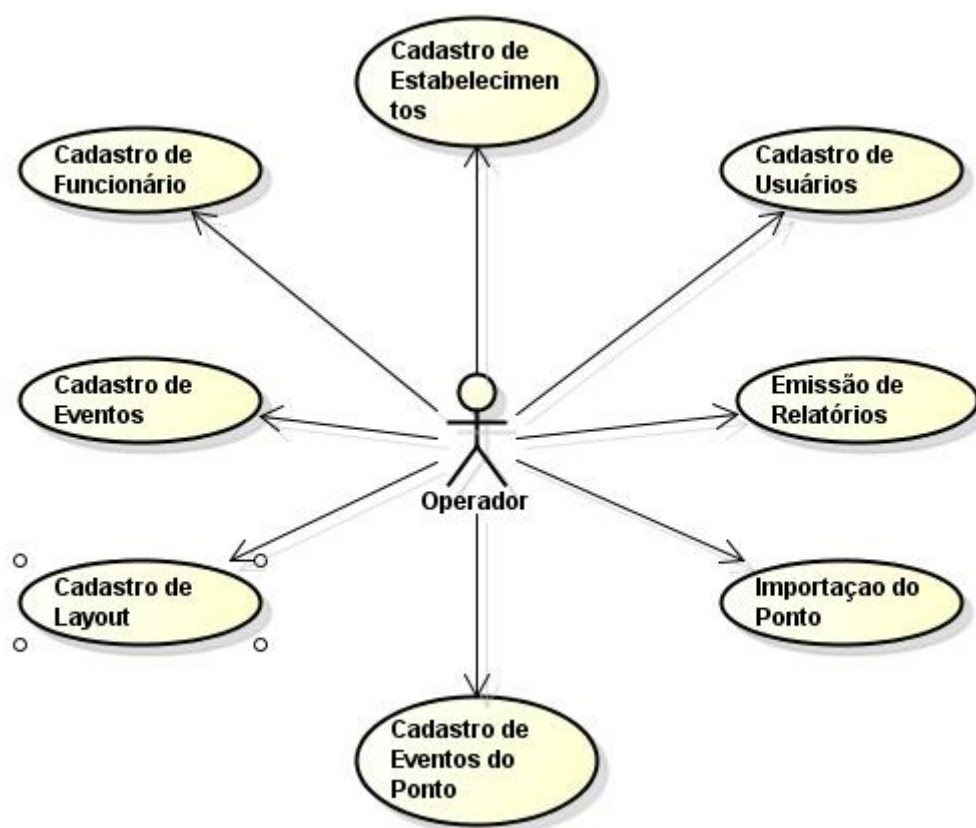


Figura 12 – Diagrama de caso de uso nível administrador

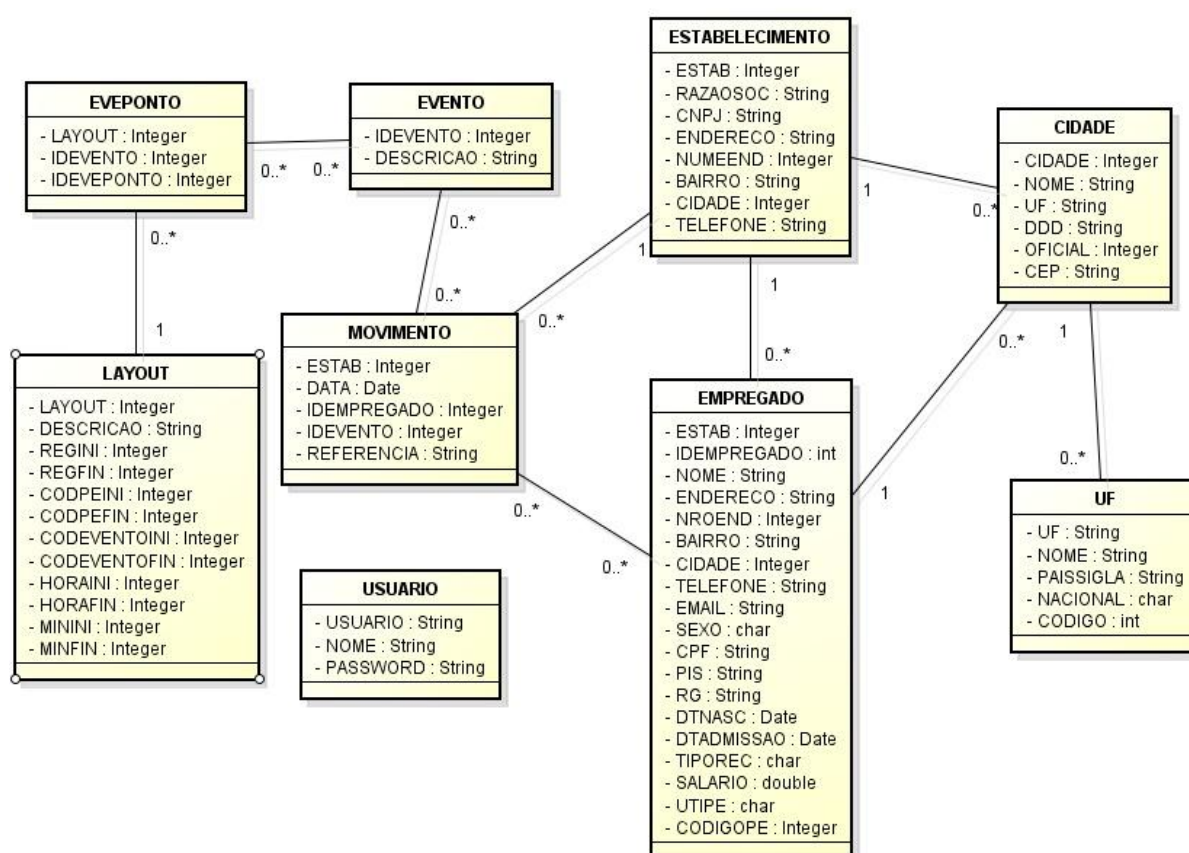
O Quadro 1 apresenta os fluxos principais dos casos de uso constantes na Figura 12. Todos os fluxos iniciam com o operador acessando o sistema.

Caso de uso	Fluxo principal
Cadastro de Funcionários	O operador selecionará na tela principal a opção para Cadastro de Funcionários. Na tela de Cadastro, o operador poderá fazer a inclusão, alteração, exclusão e pesquisa de funcionários.
Cadastro de Eventos	O operador selecionará na tela principal a opção Cadastro de Eventos. Na tela de Cadastro será possível fazer a inclusão, alteração, exclusão e pesquisa de eventos.
Cadastro de leiaute	O operador selecionará na tela principal a opção Cadastro de leiaute. Na tela de Cadastro, o operador poderá fazer a inclusão, alteração, exclusão e pesquisa de leiautes de importação.
Cadastro de Eventos do Ponto	O operador selecionará na tela principal a opção Cadastro de Eventos do Ponto. Na tela de Cadastro, o operador poderá fazer a inclusão, alteração, exclusão e pesquisa de eventos do arquivo ponto, relacionando com os eventos do sistema.
Importação do Ponto	O operador selecionará na tela principal a opção Processos Importação do ponto. O operador poderá fazer a importação do arquivo ponto.
Emissão de Relatórios	O operador selecionará na tela principal a opção Emissão de Relatórios. O sistema abrirá uma tela para que o operador possa escolher que tipo de relatório deseja emitir. O operador visualizará o relatório escolhido. O operador poderá fazer a impressão do relatório.

Cadastro de Usuários	O operador selecionará na tela principal a opção Cadastrar usuários. O sistema abrirá uma tela para que o operador possa incluir o usuário. Na tela de Cadastro, o administrador poderá fazer a inclusão, alteração, exclusão e pesquisa de usuários do sistema.
Cadastro de Estabelecimentos	O operador selecionará na tela principal a opção Cadastrar Estabelecimentos. O sistema abrirá uma tela para que o operador possa incluir o estabelecimento. Na tela de Cadastro, o administrador poderá fazer a inclusão, alteração, exclusão e pesquisa de estabelecimentos do sistema.

**Quadro 1 – Fluxo principal dos casos de uso**

A Figura 13 contém o diagrama de entidades e relacionamentos do sistema.



**Figura 13 – Diagrama de entidades e relacionamentos do sistema**

As tabelas representadas na Figura 13 se referem às classes que representam as entidades persistidas em banco de dados. A tabela EMPREGADO contém os dados do funcionário da empresa. A tabela EVENTO se refere aos eventos como, por exemplo, horas normais diurnas e horas normais noturnas. A tabela EVEPONTO contém as informações referentes aos eventos do registro de ponto. A tabela MOVIMENTO se refere às informações dos eventos importados do arquivo. A tabela LAYOUT é responsável pelas informações do leiaute do arquivo de importação. A tabela USUARIO representa os usuários do sistema. A tabela



ESTABELECIMENTO contém os dados cadastrais da empresa. E as tabelas CIDADE e UF contém informações utilizadas no cadastro de endereço de empregado e estabelecimento.

As Figuras de 14 até 18 apresentam os principais diagramas de atividades existentes no sistema. Nesses diagramas, a ordem de realização dos processos, mais especificamente de execução das atividades desses processos, pode ser observada.

A Figura 14 mostra o diagrama de atividades para inclusão de usuários.

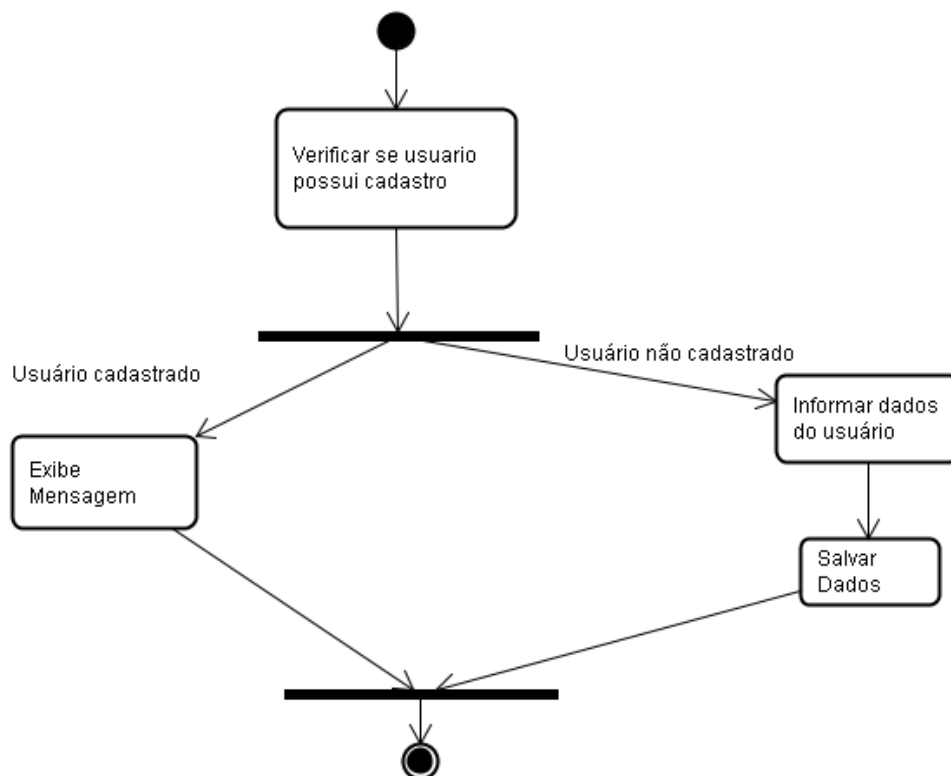
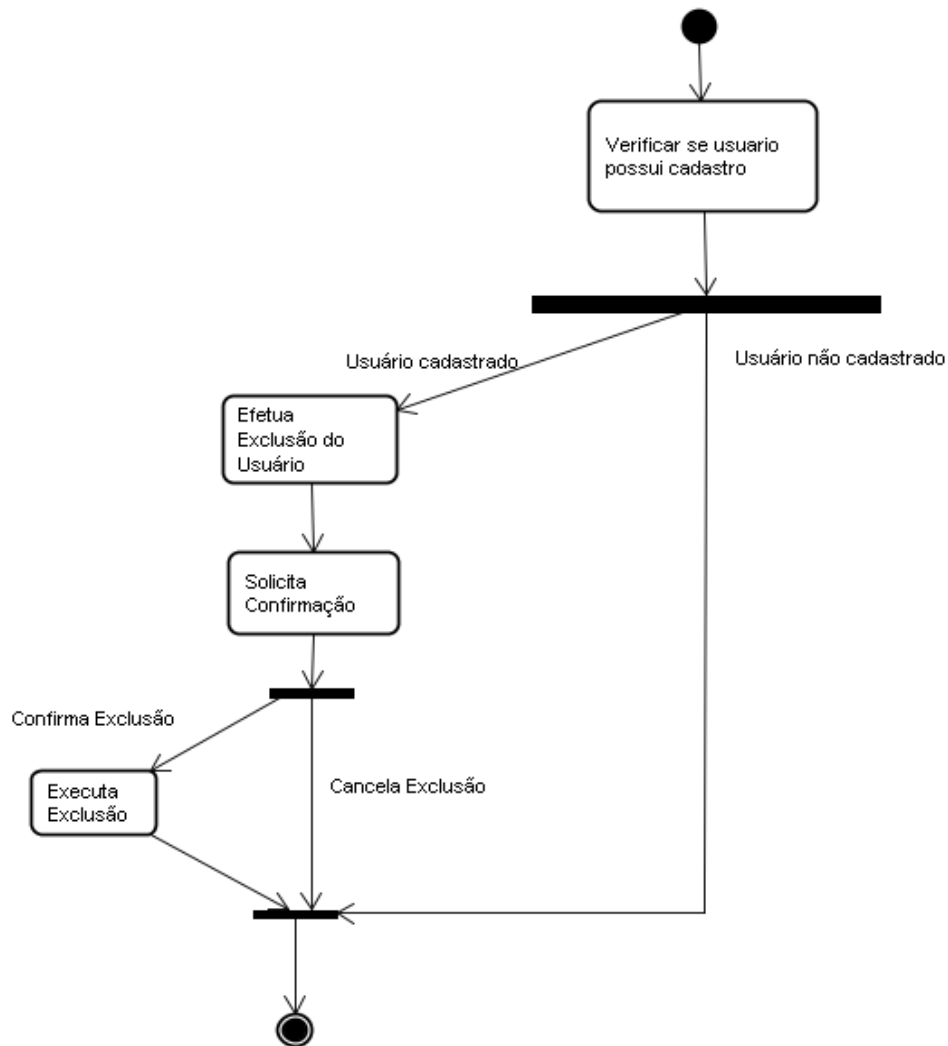


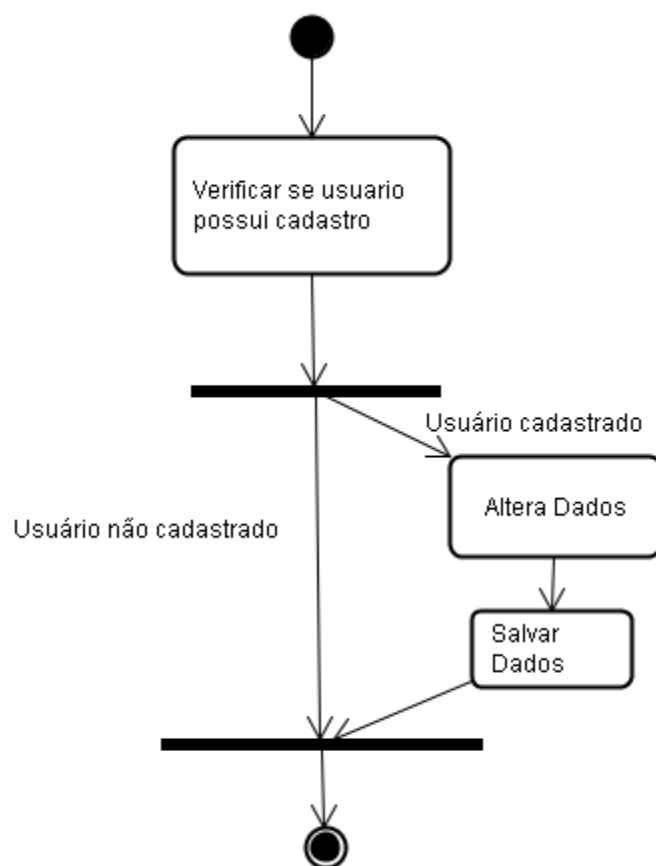
Figura 14 – Diagrama de atividade inclusão de usuário

A Figura 15 se refere ao diagrama de atividades para alteração de dados de usuários.



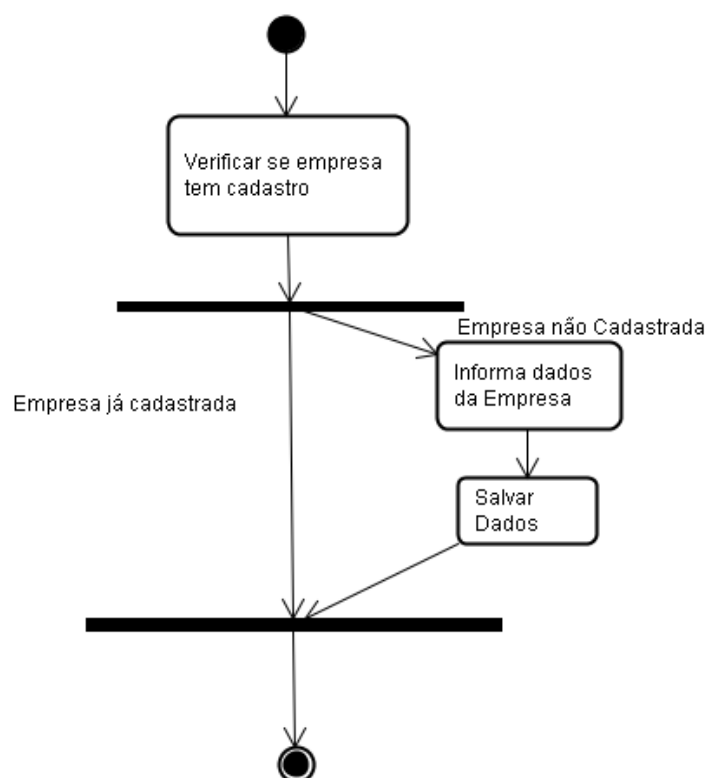
**Figura 15 – Diagrama de atividade alteração de usuário**

Na Figura 16 está o diagrama de atividades para exclusão de usuários.



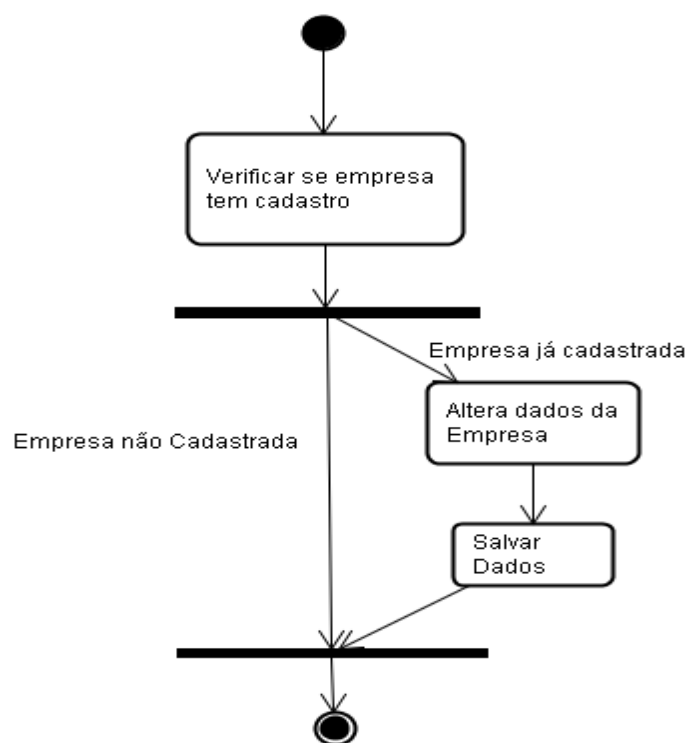
**Figura 16 – Diagrama de atividade exclusão de usuário**

A Figura 17 mostra o diagrama de atividades para inclusão da empresa.



**Figura 17 – Diagrama de atividade inclusão da empresa**

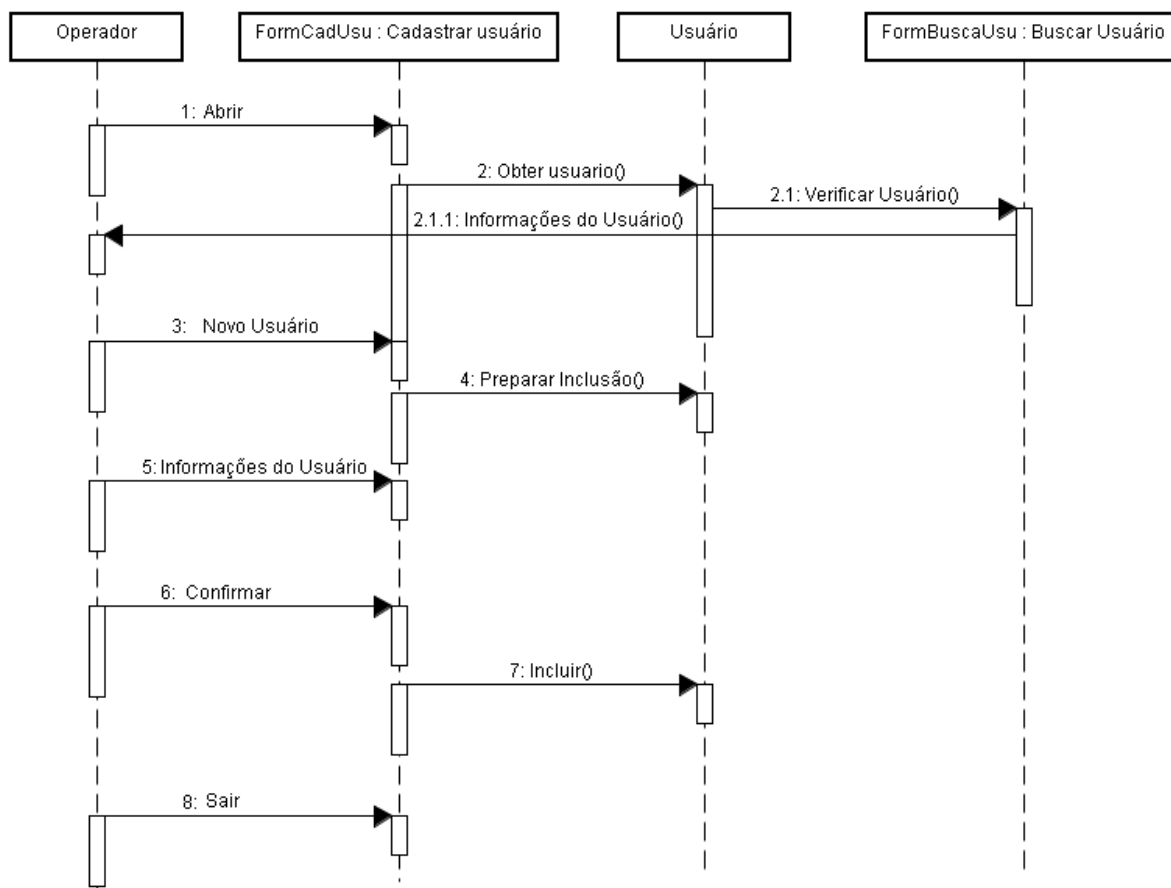
A Figura 18 mostra o diagrama de atividades para alteração da empresa.



**Figura 18 – Diagrama de atividade alteração de empresa**

As Figuras de 19 até 22 apresentam diagramas de seqüência definidos para a modelagem do sistema. Por meio desses diagramas é possível visualizar a ordem temporal em que os processos são executados.

Na Figura 19 está o diagrama de seqüência para inclusão de usuários.



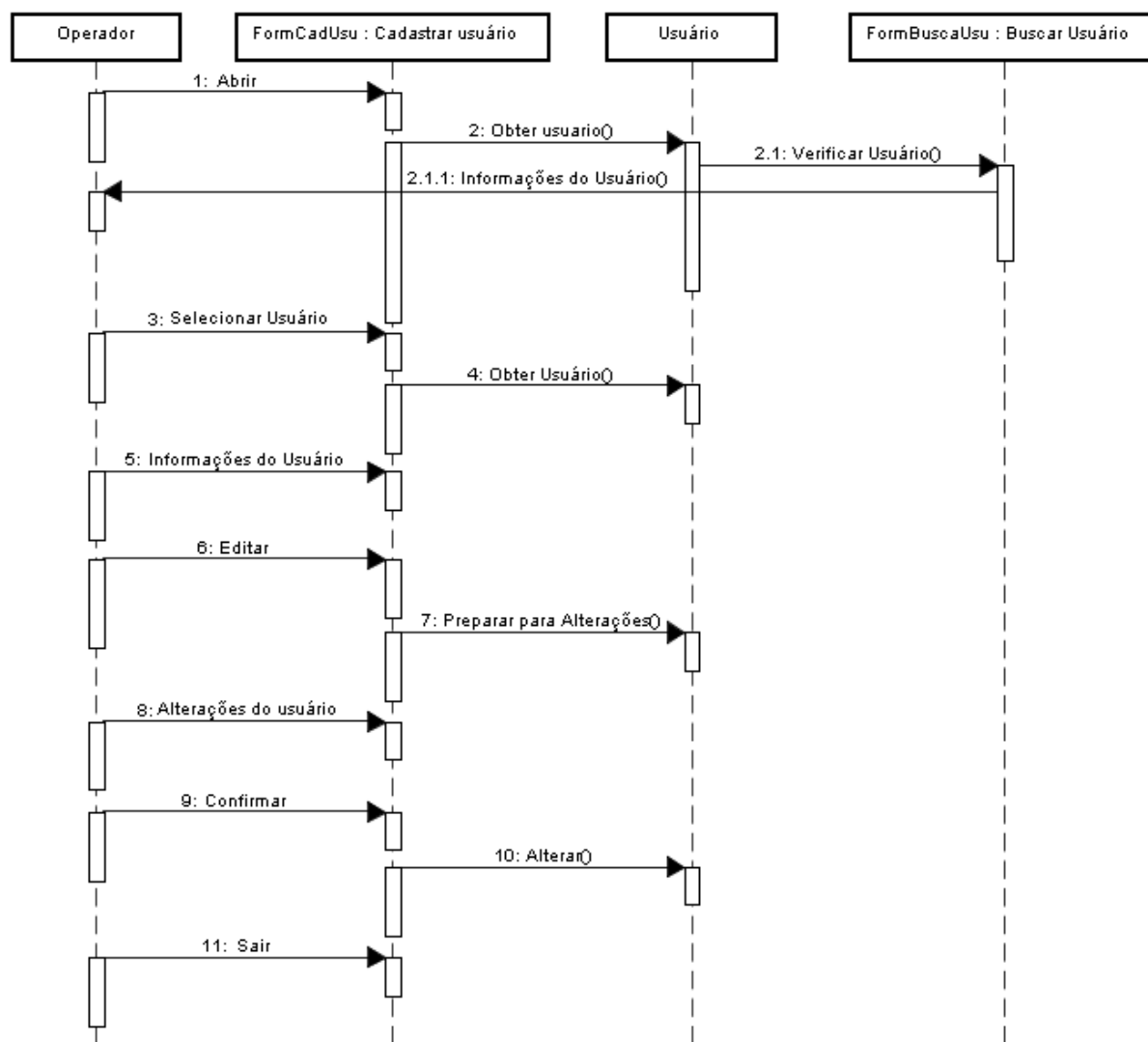
**Figura 19 – Diagrama de seqüência inclusão de usuário**

O diagrama da Figura 19 pode ser lido da seguinte maneira:

- 1) O operador solicita ao sistema a abertura da interface de cadastro de usuários.
- 2) O sistema abre a interface de cadastro de usuários.
- 3) O operador verifica se já existe o cadastro do usuário.
- 4) Após a verificação, em não havendo o respectivo cadastro, o operador retorna à interface de Cadastro de Usuários e efetiva o cadastro.
- 5) O operador informa à interface a confirmação da inclusão.
- 6) O sistema efetua a inclusão do novo usuário.
- 7) O operador fecha a interface de cadastro de usuários e retorna à tela principal.

Na Figura 20 está o diagrama de seqüência para a alteração de dados

cadastrais de um usuário.

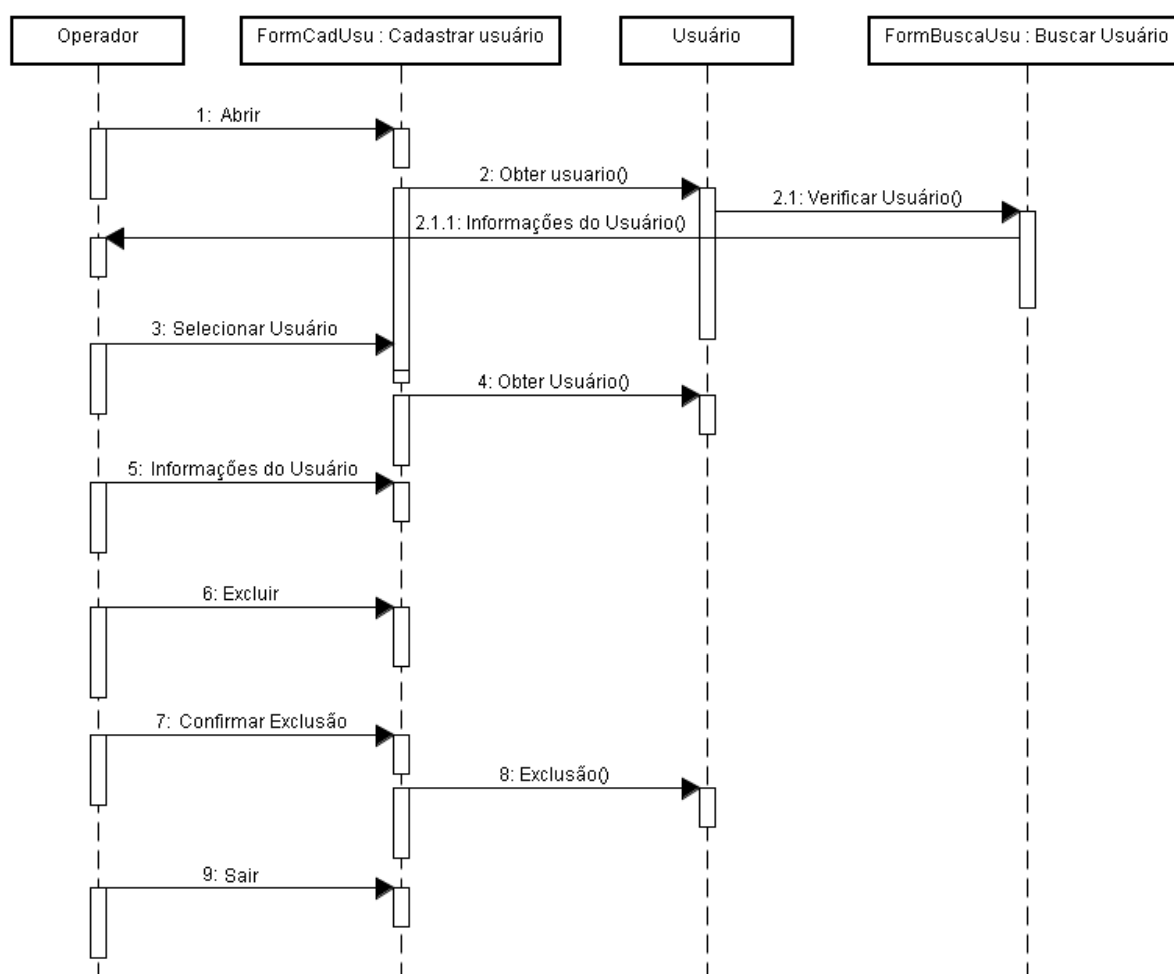


**Figura 20 – Diagrama de sequência alteração de usuário**

O diagrama da Figura 20 pode ser lido da seguinte maneira:

- 1) O operador solicita ao sistema a abertura da interface de cadastro de usuários.
- 2) O sistema abre a interface de cadastro de usuários.
- 3) O operador seleciona o usuário desejado.
- 4) O operador faz as alterações necessárias.
- 5) O operador confirma as alterações.
- 7) O sistema efetua as alterações no cadastro do usuário.
- 8) O operador fecha a interface de cadastro de usuários e retorna a tela principal.

Na Figura 21 está o diagrama de atividades para a exclusão de usuários.

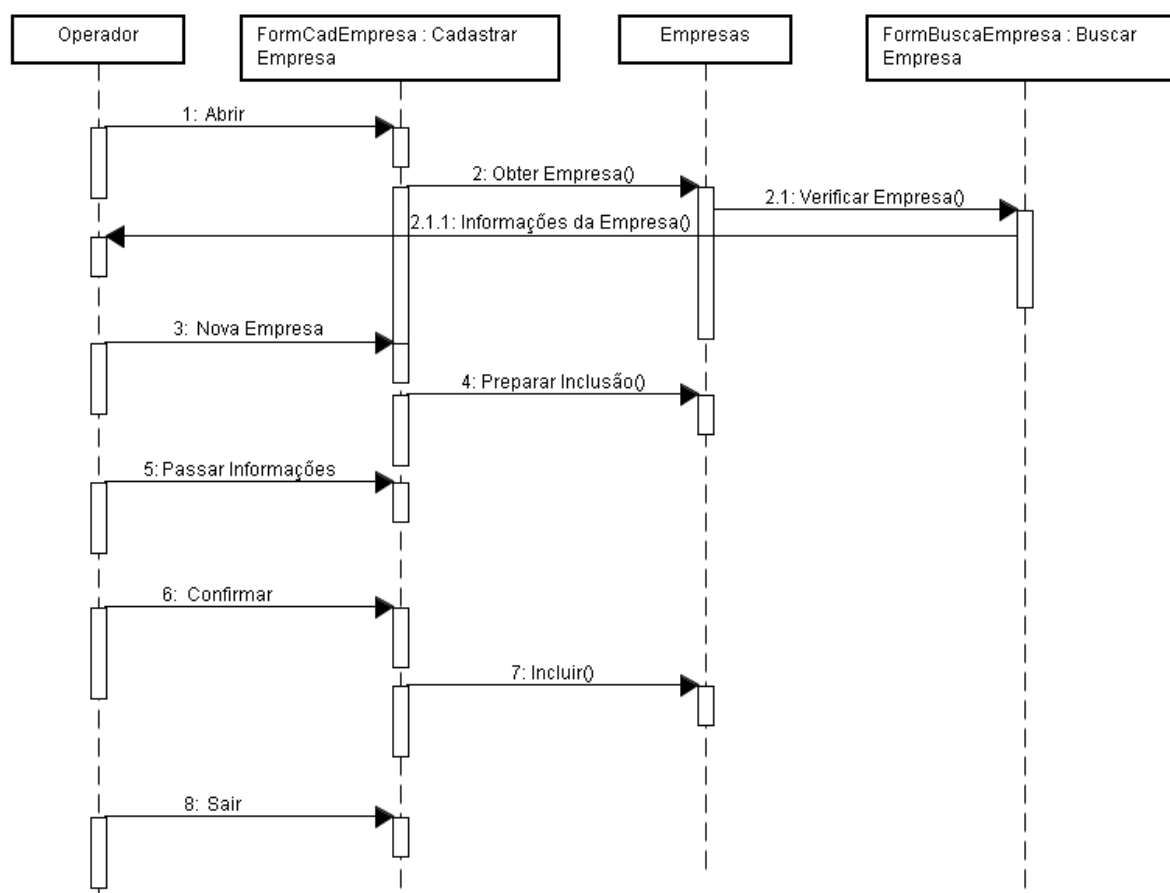


**Figura 21 – Diagrama de sequência exclusão de usuário**

O diagrama da Figura 21 pode ser lido da seguinte maneira:

- 1) O operador solicita ao sistema a abertura da interface de cadastro de usuários.
- 2) O sistema abre a interface de cadastro de usuários.
- 3) O operador seleciona o usuário desejado.
- 4) O operador solicita a exclusão do cadastro do usuário.
- 5) O sistema pede confirmação da exclusão.
- 6) O operador confirma a exclusão.
- 7) O sistema efetua a exclusão do cadastro do usuário.
- 8) O operador fecha a interface de cadastro de usuários e retorna a tela principal.

A sequência para a inclusão ou cadastro de estabelecimento é apresentada na Figura 22.



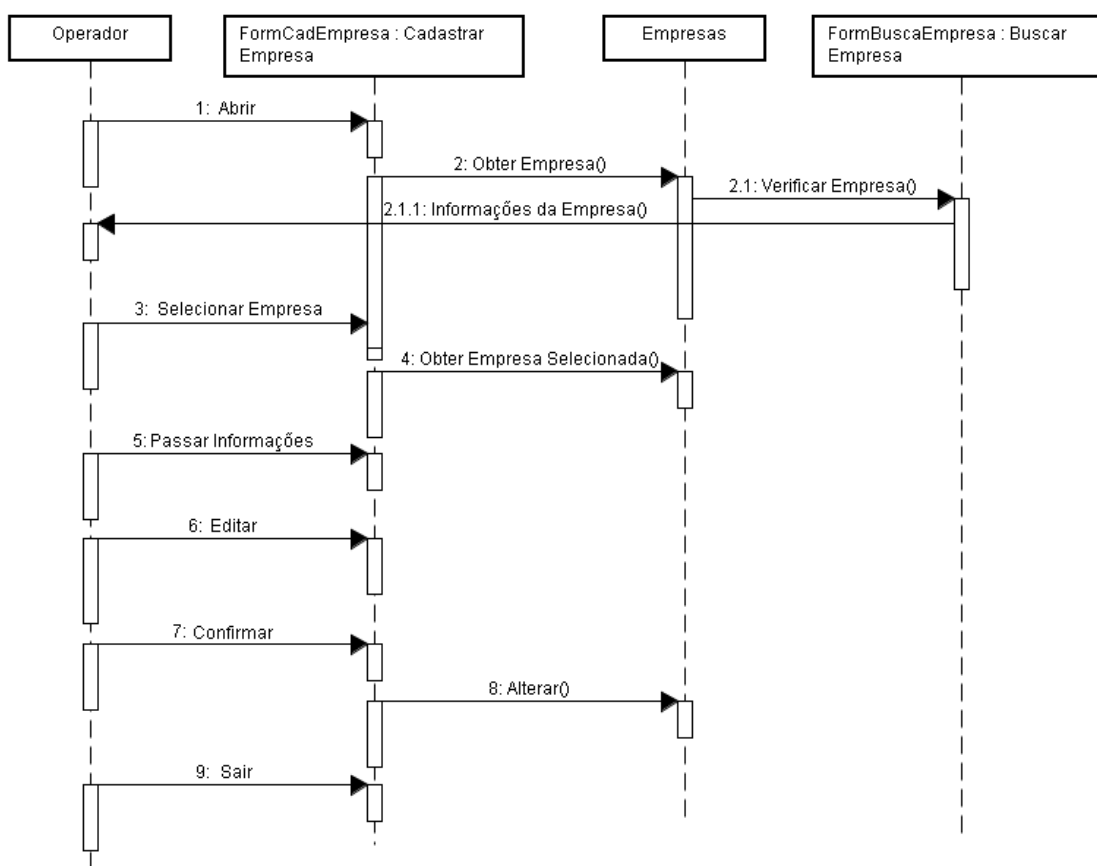
**Figura 22 – Diagrama de sequência inclusão de empresa**

O diagrama da Figura 22 pode ser lido da seguinte maneira:

- 1) O operador solicita ao sistema a abertura da interface de cadastro da empresa.
- 2) O sistema abre a interface de cadastro da empresa.
- 3) O operador verifica se já existe empresa cadastrada.
- 4) Após a verificação o operador retorna à interface de cadastro da empresa e efetiva o cadastro.
- 5) O operador informa à interface confirmação da inclusão.
- 6) O sistema efetua a inclusão da empresa.
- 7) O operador fecha a interface de cadastro de empresa e retorna a tela principal.

A Figura 23 apresenta o diagrama de sequência para a alteração dos dados cadastrais de uma empresa.





**Figura 23 – Diagrama de sequência alteração de empresa**

O diagrama da Figura 23 pode ser lido da seguinte maneira:

- 1) O operador solicita ao sistema a abertura da interface de cadastro de empresa.
- 2) O sistema abre a interface de cadastro de empresa.
- 3) O operador verifica se já existe empresa cadastrada.
- 4) O operador seleciona a empresa.
- 5) O operador faz as alterações necessárias.
- 6) O operador confirma as alterações.
- 7) O sistema efetua as alterações no cadastro da empresa.
- 8) O operador fecha a interface de cadastro da empresa e retorna a tela principal.

O Quadro 2 apresenta as interfaces de usuário definidas para o sistema.

Número	Nome	Ator	Caso de Uso	Descrição
01	Tela de Login de Acesso	Operador	Início da utilização do sistema	Interface para inicialização do uso do sistema
02	Tela Principal	Operador	Escolha do módulo a ser utilizado	Interface para escolha que qual módulo deseja-se utilizar
03	Tela de Cadastro de Estabelecimento	Operador	Cadastrar um novo estabelecimento	Interface para inclusão, alteração e exclusão de estabelecimentos
04	Tela de Cadastro de Eventos	Operador	Cadastrar um novo evento	Interface para inclusão, alteração e exclusão de eventos
05	Tela de Cadastro de UF	Operador	Cadastrar novas UF	Interface para inclusão, alteração e exclusão de UF
06	Tela para Cadastro de Cidades	Operador	Cadastrar cidades	Interface para inclusão, alteração e exclusão de cidades
07	Tela para Cadastro de Eventos do Ponto	Operador	Cadastrar eventos do ponto	Interface para inclusão, alteração e exclusão eventos do ponto
08	Tela para cadastro de leiaute	Operador	Cadastrar leiautes	Interface para inclusão, alteração e exclusão leiautes de importação.
09	Tela para Cadastro da Empregados	Operador	Cadastrar os empregados	Interface para inclusão e alteração dos empregados.
10	Tela para Emissão de Relatórios	Operador	Levantamento de Dados Gerenciais	Interface para emissão de relatórios de acompanhamento gerencial
11	Tela de Importação de Ponto	Operador	Importação do Arquivo Ponto	Interface para importação do arquivo ponto

**Quadro 2 – Interfaces de usuário**

O Quadro 3 apresenta as principais funções do sistema.

Número	Caso de Uso	Descrição
01	Cadastro de Funcionários	Cadastro e controle de funcionários do sistema
02	Cadastro de Eventos	Cadastro dos eventos que utilizará o sistema
03	Controle de Layout	Cadastro de leiautes de importação do sistema.
04	Importa ponto	Processo de importação do ponto.
05	Emissão de Relatórios	Emissão dos Relatórios do sistema

**Quadro 3 – Funções do sistema**

O Quadro 4 apresenta as principais restrições do sistema.

Número	Restrição	Descrição
01	Ambiente	O ambiente operacional a ser utilizado é o Windows xp ou superior.
02	Ambiente	O sistema deverá operar em um Pentium III ou superior.
03	Ambiente	Será utilizada impressora a laser pra emissão dos relatórios gerenciais.
04	Segurança	O acesso ao sistema será restrito através de códigos individuais e captura de digitais.

**Quadro 4 – Restrições do sistema**

### 4.3 DESCRIÇÃO DO SISTEMA

A Figura 24 apresenta a tela de *login* do sistema. Nessa tela o usuário deverá informar o nome do servidor, o nome da conexão com o banco de dados, seu *login* e senha cadastrados. As operações feitas no sistema terão a identificação do usuário conectado.

**Seleção de Usuário**

**Nome do Servidor**  
CONFIG

**Nome da Conexão**  
VSDBRH

**Usuário (UserID)**  
ADM

**Senha**  Mascarar  
○○○ ?

Para alterar sua senha, digite a senha atual e clique no botão Alterar Senha. ==>

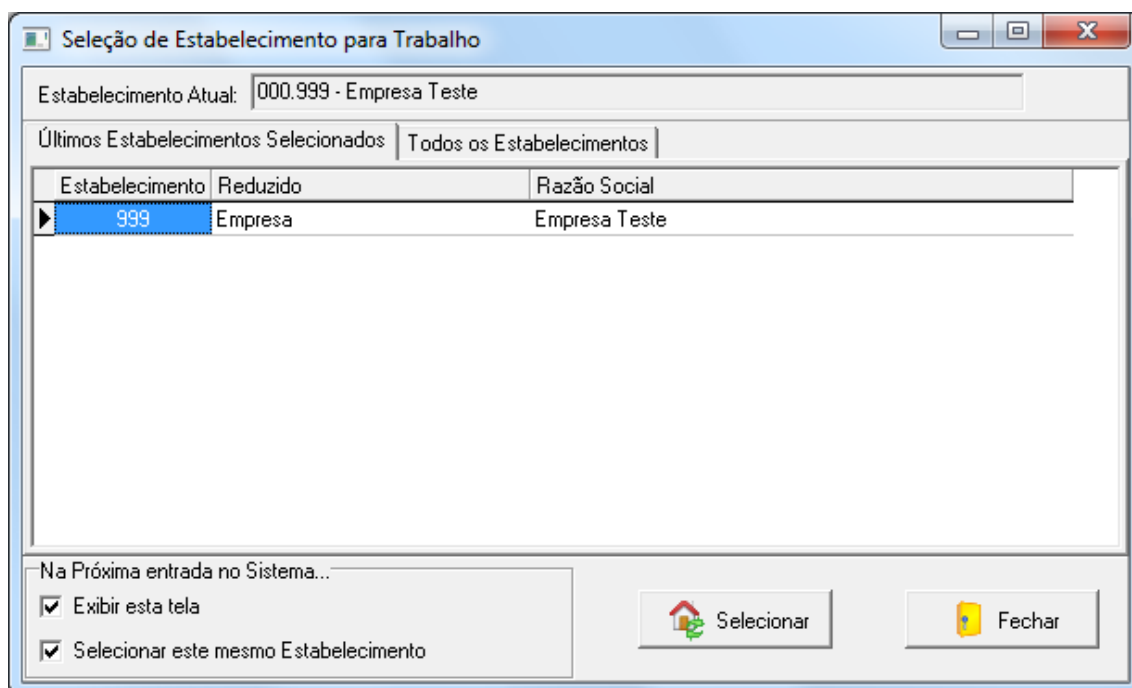
Se você é um novo usuário, deixe a senha em branco e clique diretamente em Alterar Senha.

[127.0.0.1:ViasoftServerRH.CoViasoftServerRH](http://127.0.0.1:ViasoftServerRH.CoViasoftServerRH)

Buttons: Fechar, Ok, Alterar Senha

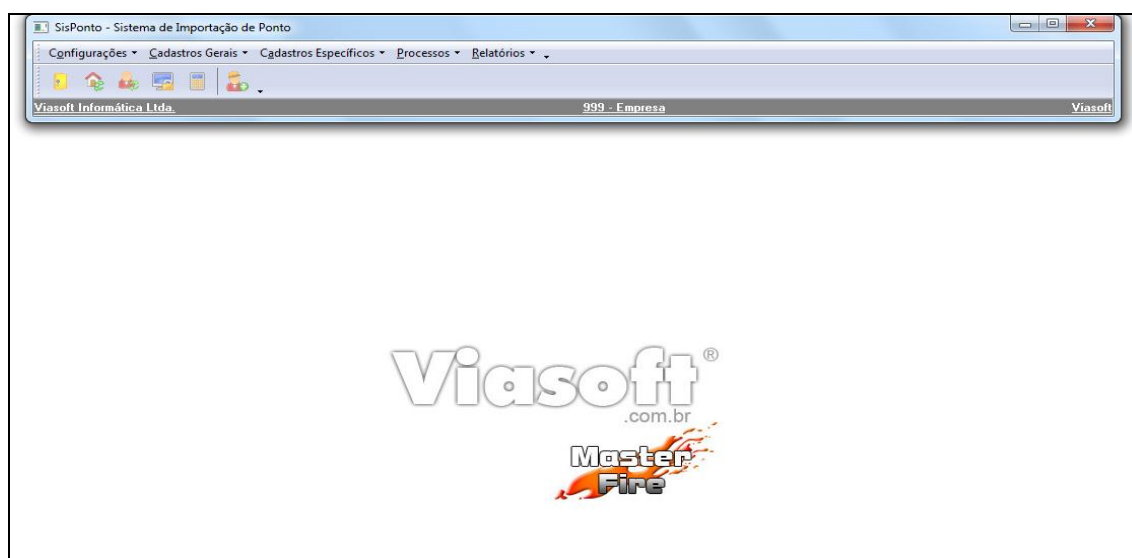
Figura 24 – Tela de *login*

Após o usuário informar a senha e logar no sistema é apresentada a tela de seleção de estabelecimento, conforme a mostra a Figura 25.



**Figura 25 – Tela de seleção do estabelecimento**

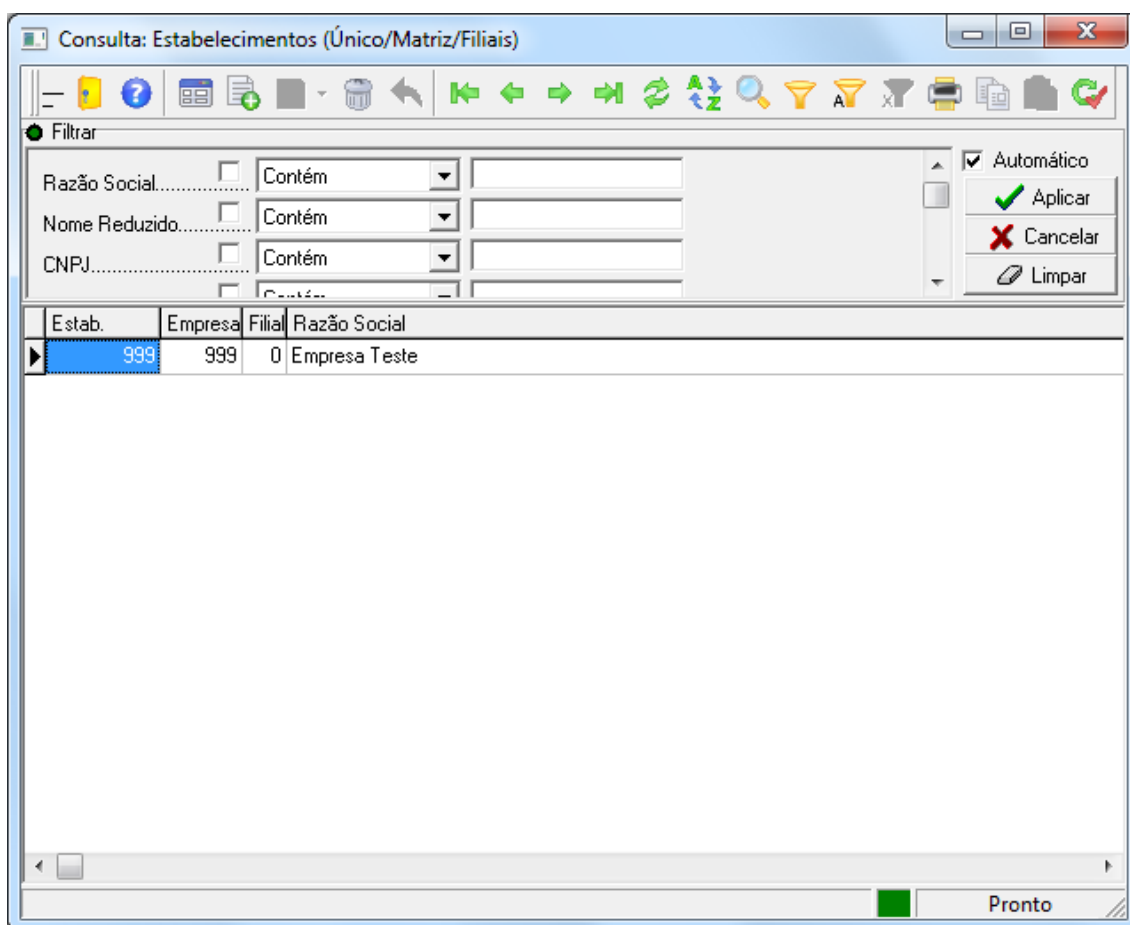
Ao selecionar o estabelecimento que deseja acessar é apresentada a tela principal do sistema, com os menus de Configurações, Cadastros Gerais, Cadastros Específicos, Processos e Relatórios, além de botões de atalhos para as principais funcionalidades do sistema. A Figura 26 apresenta a tela principal do sistema, com os menus e botões de atalho do sistema.



**Figura 26 – Tela principal do sistema**

Na Figura 27 é apresentada a tela de Manutenção de Estabelecimentos. Essa forma de pesquisa é padrão para o sistema e é apresentada em todos os seus cadastros. As opções, representadas por botões de saída da tela, inclusão,

exclusão, alteração, próximo registro, registro anterior, filtros, etc., são apresentados em todas as telas de cadastro do sistema.



**Figura 27 – Manutenção de estabelecimentos**

Ao acessar a opção de cadastro de estabelecimento (Figura 27) é apresentado um *grid* com as principais informações do cadastro de estabelecimento. Ao selecionar um estabelecimento e executar o duplo clique ou utilizar o botão de alteração, será apresentada a tela (Figura 28) para alteração do cadastro de estabelecimento.

**Figura 28 – Tela de alteração do cadastro de estabelecimento**

Os estabelecimentos cadastrados estarão disponíveis para que o usuário possa selecioná-las ao efetuar *login*.

A Figura 29 apresenta a tela de cadastro de eventos, nessa opção o usuário irá incluir todos os eventos que o sistema utilizará, como horas normais, horas extras, faltas. A descrição desses eventos será mostrada no relatório de Eventos do Ponto.

**Figura 29 – Tela de cadastro de eventos**

A Figura 30 apresenta a tela de relação dos eventos do sistema com os

eventos do ponto, pois o evento 1 do arquivo ponto pode ser o evento 100 do sistema. Assim, quando o sistema fizer a importação ele verificará nesse cadastro qual evento do sistema deve ser associado ao evento do ponto.

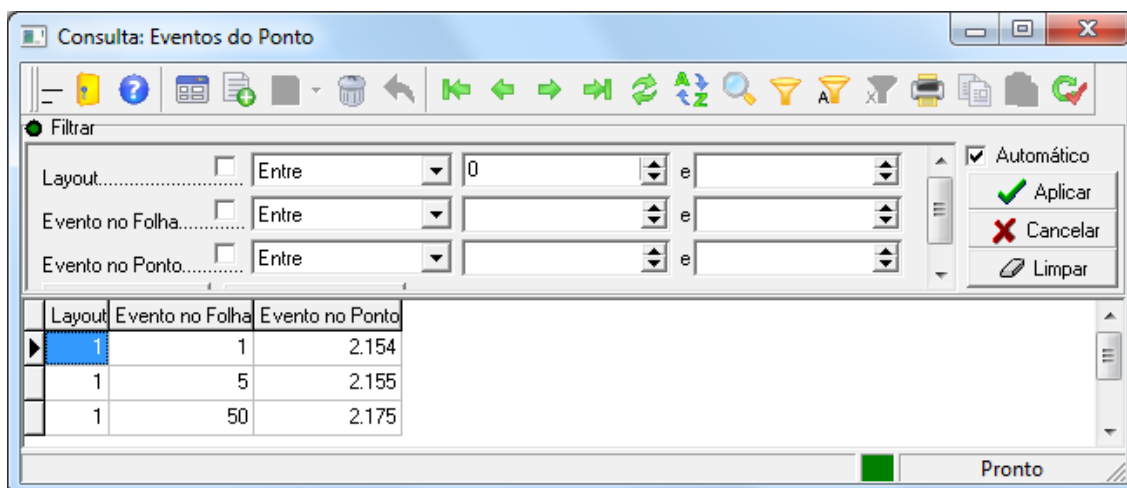


Figura 30 – Tela de cadastro de eventos do sistema do ponto

A Figura 31 apresenta o cadastro de leiaute de importação do ponto. Nesse cadastro o usuário irá informar o modo como o arquivo ponto é estruturado, incluindo a posição inicial e final do código do ponto do empregado, posição inicial e final do evento, das horas e minutos.

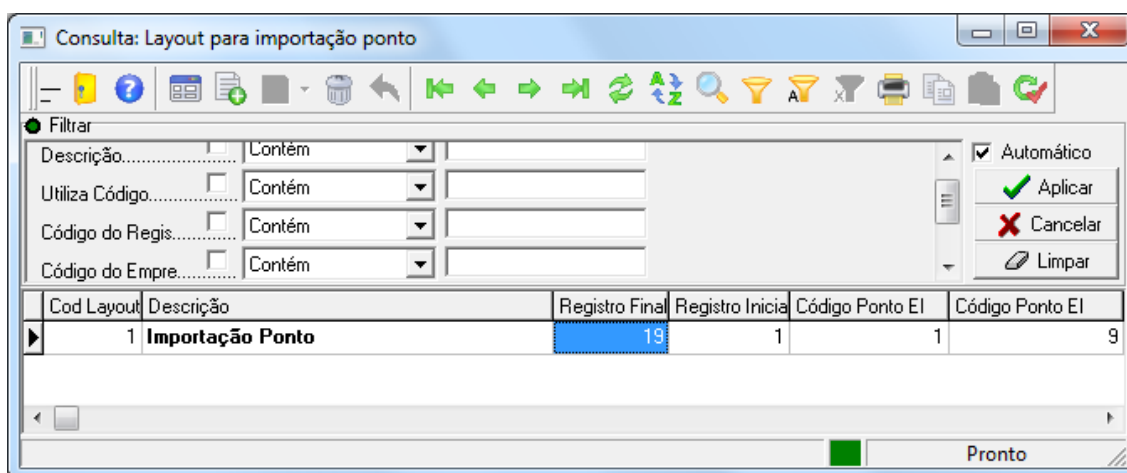
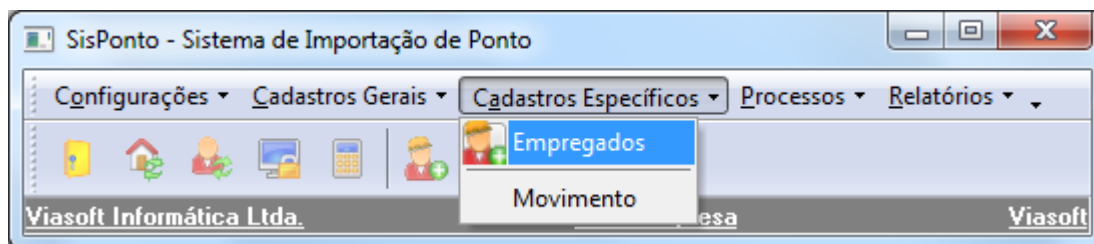


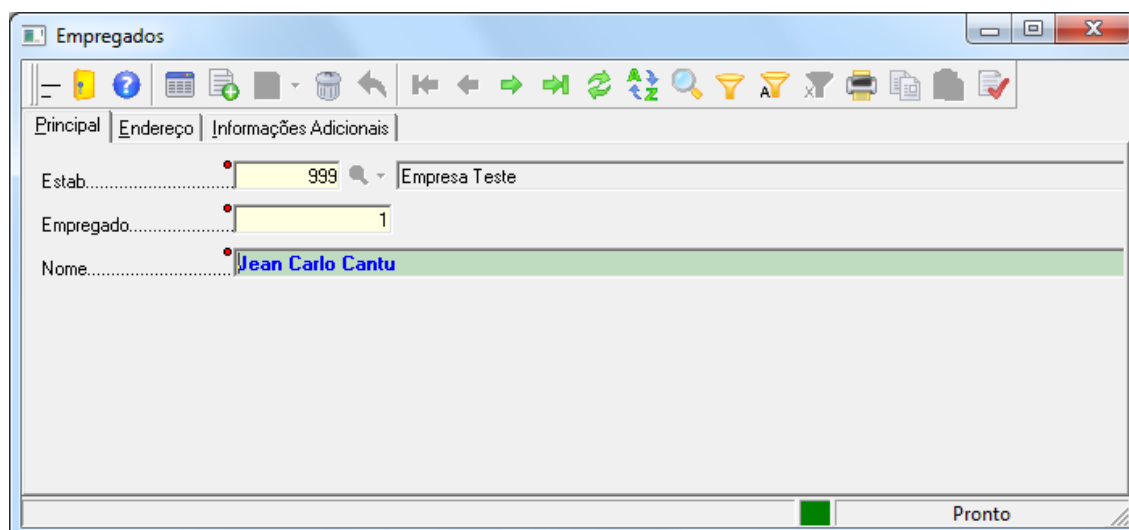
Figura 31 – Tela de cadastro de leiaute de importação

A Figura 32 apresenta os itens do menu cadastros específicos que são Empregados e Movimento. Em Empregados são cadastrados os empregados do estabelecimento selecionado e em Movimento são cadastrados os eventos que são importados do arquivo de ponto.



**Figura 32 – Itens do menu cadastros específicos**

A Figura 33 apresenta a tela de manutenção de empregados. Essa tela é composta pelas abas Principal, Endereço e Informações Adicionais, além dos botões padrões de inclusão, exclusão, alteração, *refresh*, etc, utilizados em todos os cadastros do sistema. Na aba Principal (Figura 33) é informado o estabelecimento que o empregado está cadastrado, código do empregado e nome.



**Figura 33 – Manutenção de empregados - aba principal**

Na aba Endereço do Cadastro de Empregados (Figura 34) são informados os dados referente ao endereço do empregado, como rua, cep, número e cidade.



The screenshot shows the 'Empregados' application window with the 'Endereço' tab selected. The form contains the following fields and values:

Endereço.....	Desvaldo Aranha	
Nro.....	772	
Bairro.....	Centro	
Código da Cidade.....	03613	Pato Branco ; PR
CEP.....	85504350	
Telefone.....	91067240	
e-mail.....	jeanccantu@gmail.com	
Sexo.....	Masculino	

The status bar at the bottom right shows 'Pronto'.

**Figura 34 – Manutenção de empregados - aba endereço**

Na aba Informações Adicionais do Cadastro de Empregados (Figura 35) são informados os dados referente ao cadastro na empresa, como data de admissão, salário e código do ponto eletrônico.

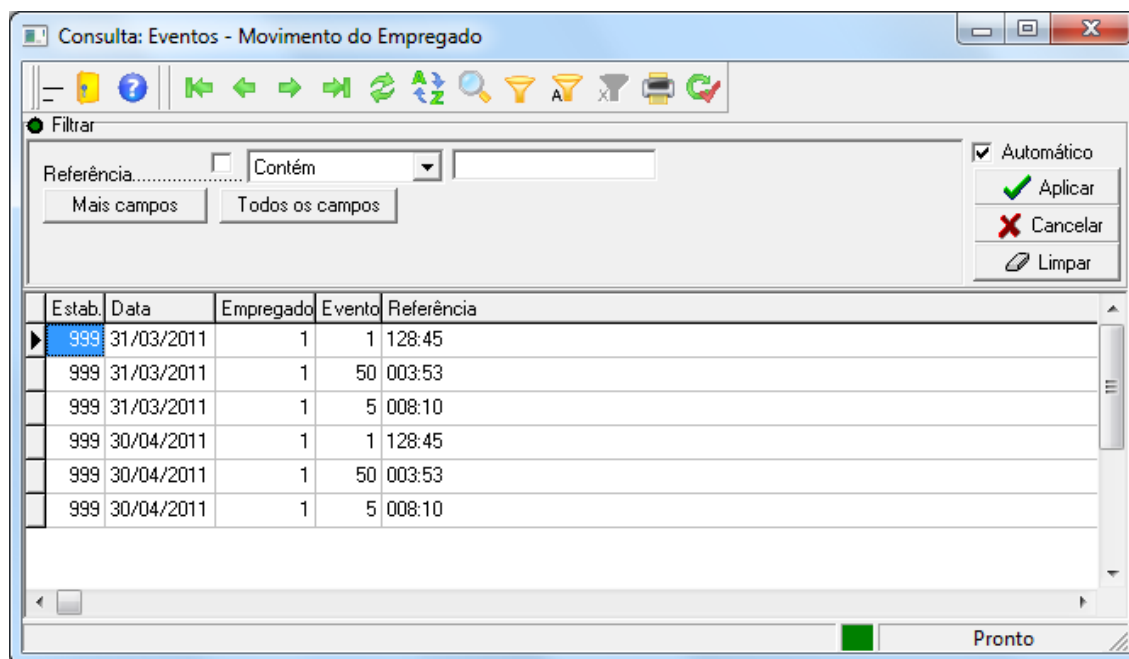
The screenshot shows the 'Empregados' application window with the 'Informações Adicionais' tab selected. The form contains the following fields and values:

CPF.....	05156095950	
Pis/Ci.....	00000000000	
RG.....	94276586	
Data Nascimento.....	27/08/1988	[Calendar Icon]
Data de Admissão.....	05/01/2011	[Calendar Icon]
Tipo do Recebimento.....	Dinheiro	
Salário.....	5.000,00	
Utiliza Ponto Eletrônico?.....	Sim	
Código Ponto Eletrônico.....	1	

The status bar at the bottom right shows 'Pronto'.

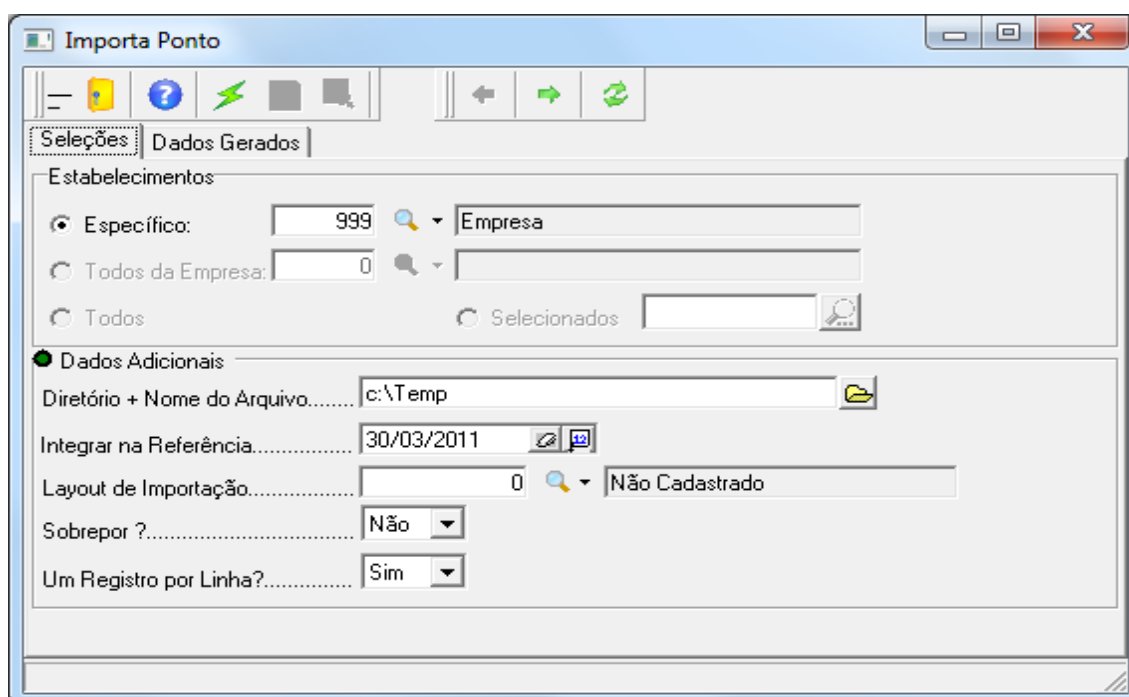
**Figura 35 – Manutenção de empregados - aba informações adicionais**

Na tela de movimento (Figura 36) é apresentado somente o *grid* com os dados importados do ponto. Isso porque nesse cadastro não há a opção de exclusão, alteração e inclusão de dados. Esses dados serão alimentados somente pelo processo de importação de ponto.



**Figura 36 – Tela de movimento**

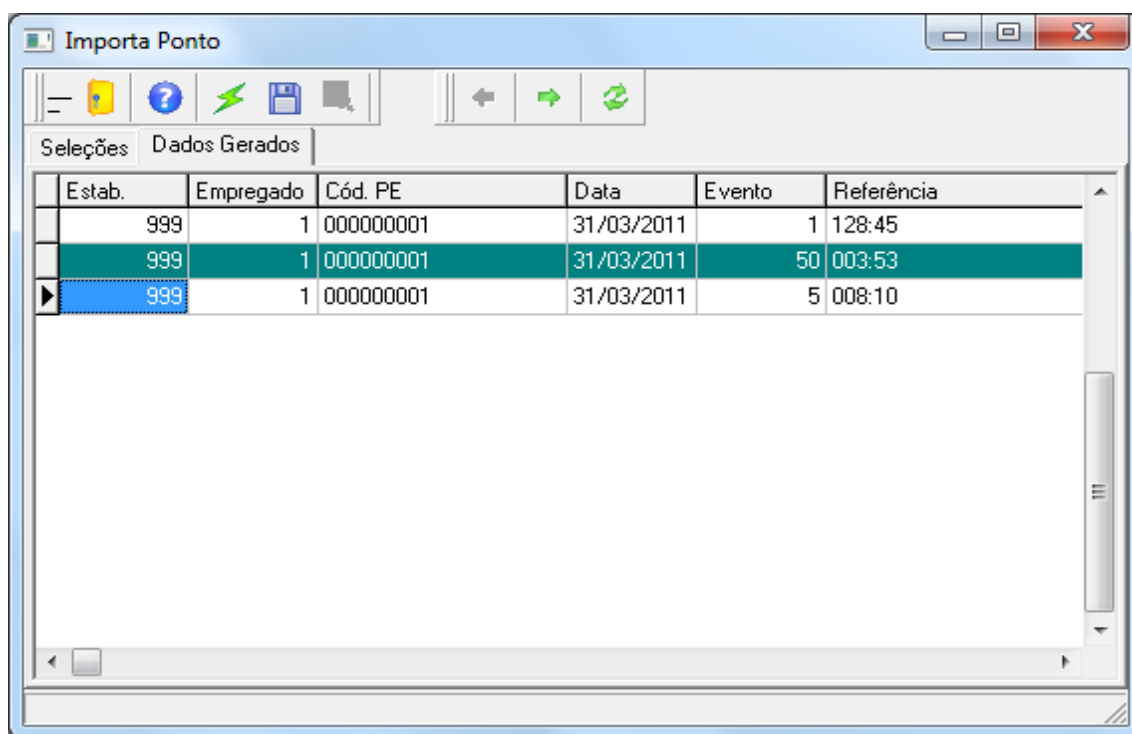
A Figura 37 apresenta a tela de Importação de Ponto. Nessa tela o usuário irá informar o estabelecimento que irá importar o ponto. E nos dados adicionais informará o caminho onde está localizado o arquivo ponto, a data de referência de importação do ponto, o leiaute de importação se deseja sobrepor os dados já importados e se é um registro por linha no arquivo ponto.



**Figura 37 – Importação de ponto**

Após a importação do arquivo ponto a aba de Dados Gerados (Figura 38) é

visualizada. Nessa tela são mostrados os registros importados do arquivo ponto. Após conferir os registros importados o usuário pode salvá-los.

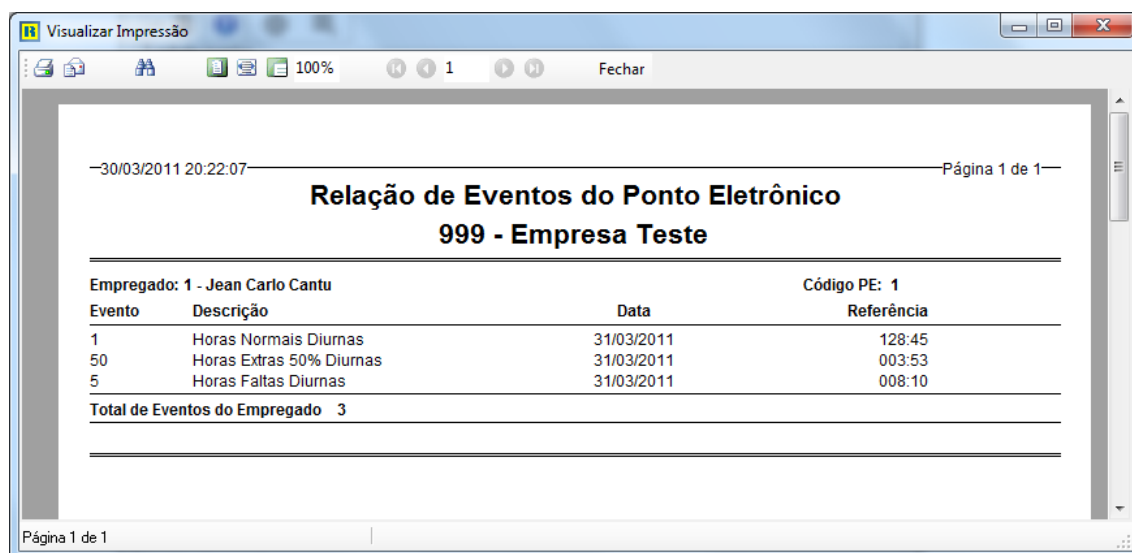


The screenshot shows a window titled 'Importa Ponto' with a toolbar and two tabs: 'Seleções' and 'Dados Gerados'. The 'Dados Gerados' tab is active, displaying a table with the following data:

Estab.	Empregado	Cód. PE	Data	Evento	Referência
999	1	000000001	31/03/2011	1	128:45
999	1	000000001	31/03/2011	50	003:53
999	1	000000001	31/03/2011	5	008:10

Figura 38 – Tela de dados gerados (importação do ponto)

Por meio dos dados importados é emitido o relatório de Eventos do Ponto (Figura 39).



The screenshot shows a window titled 'Visualizar Impressão' displaying a report. The report header includes the date and time '-30/03/2011 20:22:07' and 'Página 1 de 1'. The main title is 'Relação de Eventos do Ponto Eletrônico 999 - Empresa Teste'. Below the title, the employee information is shown: 'Empregado: 1 - Jean Carlo Cantu' and 'Código PE: 1'. The report contains a table of events:

Evento	Descrição	Data	Referência
1	Horas Normais Diurnas	31/03/2011	128:45
50	Horas Extras 50% Diurnas	31/03/2011	003:53
5	Horas Faltas Diurnas	31/03/2011	008:10

At the bottom of the table, it states: 'Total de Eventos do Empregado 3'. The footer of the window shows 'Página 1 de 1'.

Figura 39 – Relatório de eventos do ponto

#### 4.4 IMPLEMENTAÇÃO DO SISTEMA

Nesta seção são apresentados códigos com o objetivo de exemplificar a implementação de um sistema utilizando Borland Delphi 7 e Report Builder.

A Listagem 1 apresenta uma parte do código do cadastro padrão do sistema. Os cadastros implementados no sistema herdarão desse cadastro padrão. Assim, as funcionalidades principais como, inclusão, exclusão, alteração não precisarão ser implementadas em cada cadastro e sim implementado uma única vez nesse cadastro padrão.

```

procedure TFormEspCds.ActSalvarExecute(Sender: TObject);
var
  i: integer;
begin
  OperacaoDoForm := tomValidando;
  try
    for i := 0 to High(DadosColuna) do
      begin
        ValidaComponente( DadosColuna[i].oComp );
      end;
    end;
    CheckFinal;
    if UListErro.ErrosDoForm(self) = 0 then {Todos estao Ok}
      begin
        if lTemPersonalizacao then
          SBPer.ValidaTudo;
        if UListErro.ErrosDoForm(self) > 0 then
          begin
            ActMostraErros.OnExecute(ActMostraErros);
            Abort; {Aborta possivel saida do Form}
          end;

        //Nenhum erro nas validacoes, verifica os warnings
        if ( UListErro.WarningsDoForm( self ) = 0 ) or
           UListErro.MostraWarningsDoForm(self, PosicionaNoErro ) then
          begin
            OperacaoDoForm := tomGravando;

            inherited; {Grava}

            EstadoDosCamposChave(True); {Protege chaves}
            if lTemPersonalizacao then
              SBPer.ChaveProtegida := True; {Protege chaves}
            //FocaPrimeiroComponente; {Nao usar}
          end;
        end
      else {Mostra a tela de erros}
        begin
          ActMostraErros.OnExecute(ActMostraErros);
          Abort; {Aborta possivel saida do Form}
        end;
      end;
    end;
  end;
end;

```

Listagem 1 - Código da *unit* de cadastro padrão

A Listagem 2 apresenta parte do código do processo padrão. Os processos que forem implementados no projeto poderão herdar dessa *unit*. Assim, já estarão implementadas as funcionalidades de filtros, botões padrões de processar, salvar e fechar. E terá a aba de dados gerados, restando ao programador alimentar esses dados gerados conforme é executado o processo.

```
procedure TFRhProcPadrao.SalvaRegistros(cds: TVsClientDataSet;
  msg: string);
begin
  try
    try
      //cdsSalarios.ApplyUpdates(-1);
      dmConexao.CDSApplyUpdates([cds]);
      bResp.RespOk('Processo Concluído!', [msg]);
      actSalvar.Enabled := False;
    except
      bResp.RespCancel('Atenção!', ['Erro encontrado, Processo Cancelado!']);
      actSalvar.Enabled := true; // alterado para true porque o usuario pode
                                // modificando informacoes no grid somente.
    end;
  finally
    EmProcesso := False;
  end;
end;
```

Listagem 2 - Código da unit de processo padrão

A Figura 40 apresenta o modelo de relatório padrão do sistema. Os relatórios desenvolvidos poderão herdar desse relatório padrão. Esse relatório já possui filtro de estabelecimentos e intervalos de datas padrão e empregados, não sendo necessário implementar em cada relatório desenvolvido.

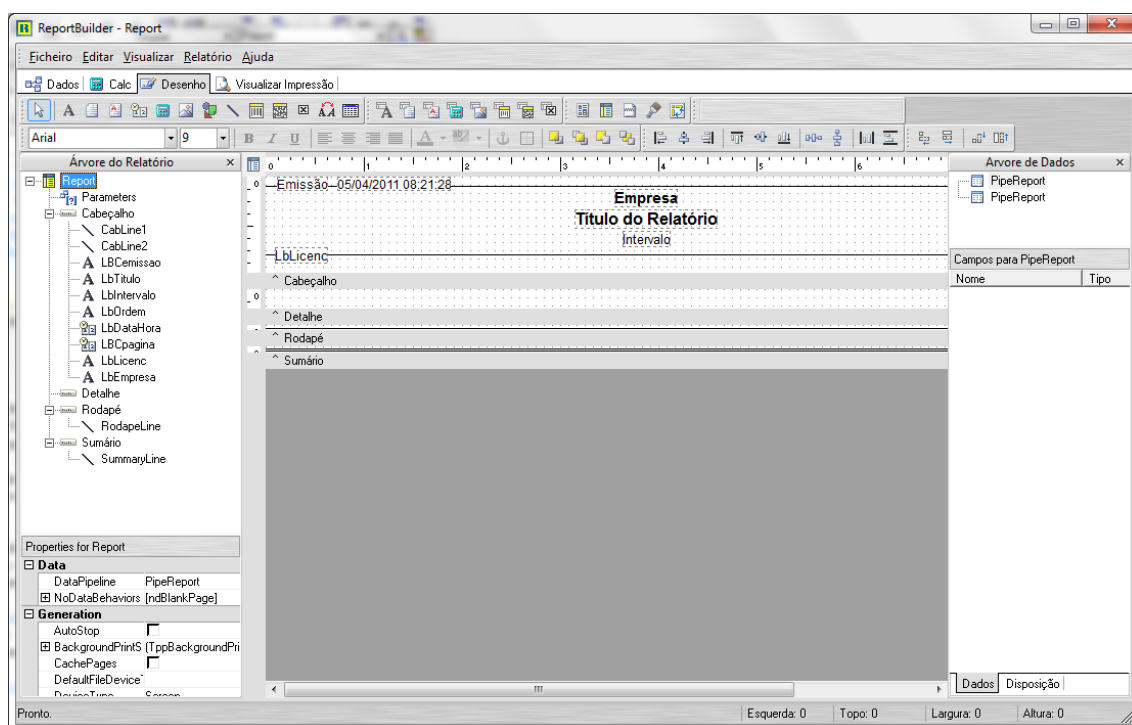


Figura 40 – Modelo de relatório padrão

A Listagem 3 apresenta o código da função SelPath. Essa função é utilizada no processo de importação de ponto para selecionar o diretório no qual se encontra o arquivo para importação do ponto.

```
function SelPath(const cCaption: string; out cPath: String; lFile,
                lSelFile: boolean; const aTipos: array of string): boolean;
{ cria e inicializa as propriedades do form para seleção de drive, diretório e [arquivo] }
var
  i: integer;
  OD: TOpenDialog;
  cRoot: WideString;
begin
  if (lSelFile) then //selecao de arquivo
  begin
    OD := TOpenDialog.Create(nil);
    try
      OD.Title := cCaption;
      OD.Options := [ofFileMustExist, ofEnableSizing];
      for i := 0 to High(aTipos) do
        OD.Filter := OD.Filter + aTipos[i] + '|';
      if (OD.Filter = '') then
        OD.Filter := 'Qualquer arquivo (*.*)|*.*';
      if (cPath <> '') then
        OD.InitialDir := cPath;
      Result := OD.Execute;
      if (Result) then
        cPath := OD.FileName
      else
        cPath := '';
    finally
      FreeAndNil(OD);
    end;
  end
  else //Selecao de pasta
  begin
    cRoot := cPath;
    Result := FileCtrl.SelectDirectory(cCaption, cRoot, cPath);
  end;
end;
```

Listagem 3 - Código para seleção de arquivo em um diretório

A Listagem 4 apresenta o código da função LeLinha que é utilizada na importação do ponto. Essa função lerá o arquivo ponto linha a linha para, assim, efetuar a importação.

```
function LeLinha(cArq: string; var hArq: TextFile; var cStr: string;
                lMsgErro: boolean=True): boolean;
{ le uma linha em hArq (handle) a partir da posição atual do arquivo,
  atribuindo para cStr (ReadLn) }
var
  nErro: integer;
begin
  Result := False;
  while True do
    try
      {$I-}
      ReadLn(hArq, cStr);
      nErro := IOResult;
      {$I+}
      if nErro <> 0 then
        Raise Exception.CreateFmt('Não foi possível ler o arquivo: '+ #10 +
                                   '%s' + #10 + #10 + 'Erro: %s - %s',
                                   [cArq, IntToStr(nErro),
                                    BibErrMsg.DescErroBN(nErro)] );

      Result := True;
      break;
    except
      FechaArq(hArq, False);
      if lMsgErro then
        raise
      else
        break;
    end;
  end;
end;
```

**Listagem 4 - Código da função LeLinha**

A Listagem 5 apresenta parte do código que é utilizado ao clicar em processar na importação do ponto. Nesse código é aberto o arquivo do ponto para leitura e é feita a verificação se o usuário informou se é um registro por linha no arquivo ponto ou não. Se é um registro por linha, o sistema irá chamar a função leComRegistro (Listagem 6) e se estiver todo o código na mesma linha, o sistema irá chamar a função LeLinha (Listagem 4) e posteriormente a função GravaDados (Listagem 7). Se não for possível para o sistema ler o arquivo do ponto será retornada uma mensagem ao usuário, informando que o caminho ou nome do arquivo especificado estão incorretos.





registros em uma única linha. Somente é inserido os dados no ClientDataSet cdsImportaPonto, para ser executado o applyUpdate posteriormente salvando os dados na tabela RHMOVIM.

```

procedure TFRhProcImportaPonto.GravaDados(var cStr : string);
var
  nPE : Variant; min : Double;
begin
  try
    cdsImportaPonto.Insert;
    cdsImportaPonto.FieldByName('ESTAB').Value := whEstab.Selecoes;
    cdsImportaPonto.FieldByName('DATA').Value := EB_Referencia.Data;
    cdsImportaPonto.FieldByName('CODIGOPE').Value := Copy(cStr, cdsSelecoes.AsInt('CODEPEINI'),
      (cdsSelecoes.AsInt('CODEPEFIN') -
      cdsSelecoes.AsInt('CODEPEINI')) + 1);

    if (cdsRhEvePonto.Locate('IDEVEPONTO', VarArrayOf([IntOf(Copy(cStr, cdsSelecoes.AsInt('CODEVENTOINI'),
      (cdsSelecoes.AsInt('CODEVENTOFIN') -
      cdsSelecoes.AsInt('CODEVENTOINI')) + 1)), [])) then
    begin
      cdsImportaPonto.FieldByName('IDEVENTO').Value := cdsRhEvePonto.AsInt('IDEVENTO');
    end
    else
    begin
      cdsImportaPonto.FieldByName('IDEVENTO').Value := strToInt(Copy(cStr, cdsSelecoes.AsInt('CODEVENTOINI'),
      (cdsSelecoes.AsInt('CODEVENTOFIN') -
      cdsSelecoes.AsInt('CODEVENTOINI')) + 1));
    end;

    if (cdsSelecoes.FieldByName('CODEMPN').Value = 'S') then
      nPE := StrToFloat(cdsImportaPonto.FieldByName('CODIGOPE').Value)
    else
      nPE := cdsImportaPonto.FieldByName('CODIGOPE').Value;

    cdsImportaPonto.FieldByName('IDEMPREGADO').Value := dmConexao.QueryPegaCampo('SEL_RHEMPCODPE', 'IDEMPREGADO',
      ['P', 'ESTAB', whEstab.Selecoes,
      'P', 'CODIGOPE', strOf(nPE)],
      [ftInteger, ftString], [0, 200]);
  
```

Listagem 7 - Parte do código da função GravaDados

Por meio da Figura 41 é possível verificar a tela de menu padrão.

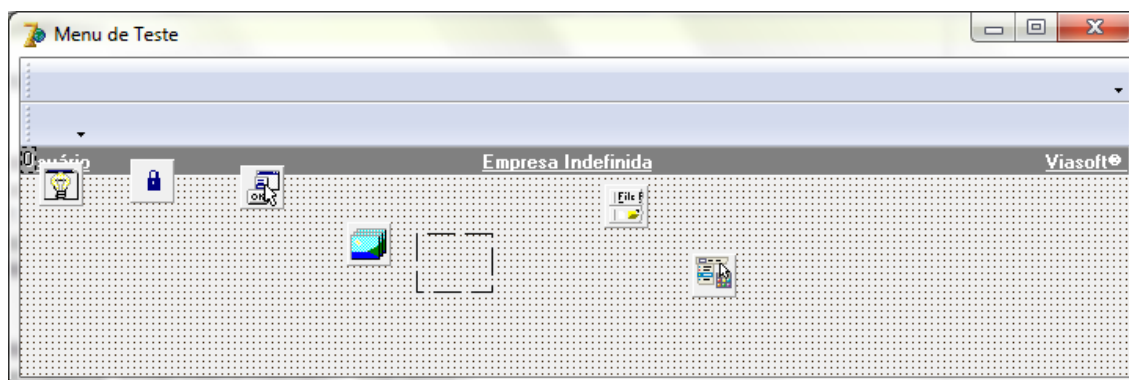


Figura 41 – Tela menu padrão

Na Listagem 8 é apresentado o código para chamada do Relatório de Ponto.

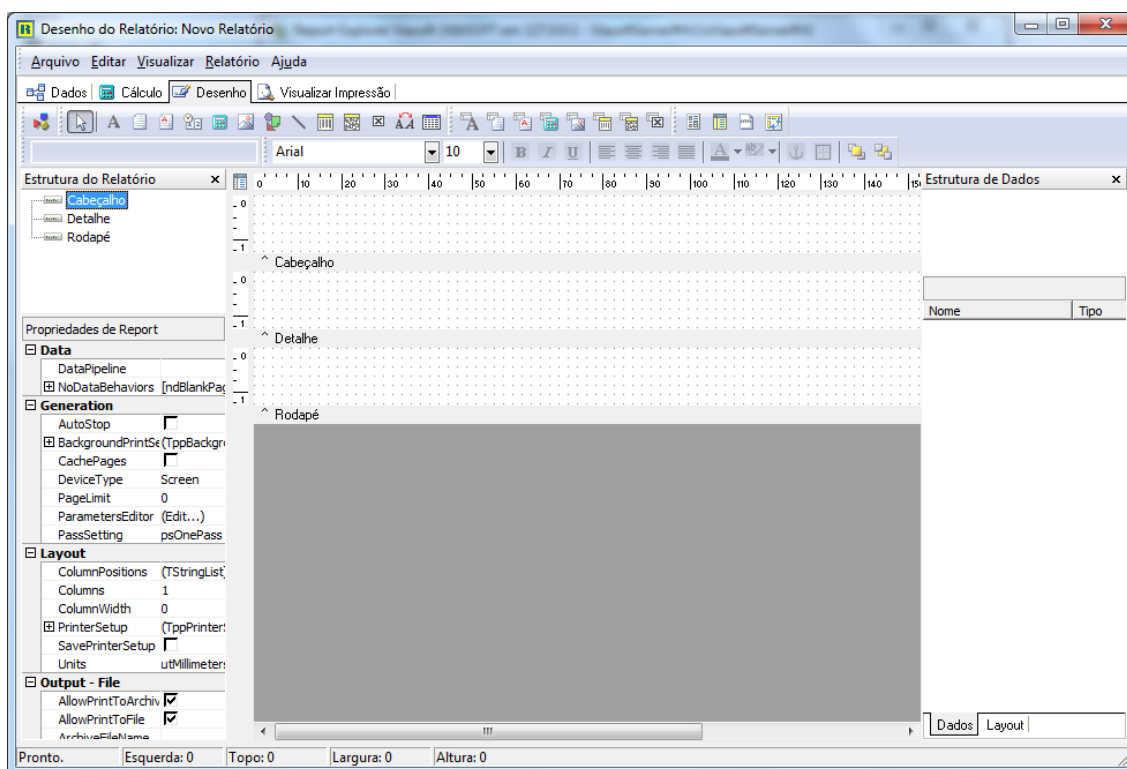
```

procedure TVsMenu.MI_RhRelPontoExecute(Sender: TObject);
begin
  VsMenu.ShowForm(TFRhRelPonto, FRhRelPonto, 'FRhRelPonto', sender);
end;

```

Listagem 8 - Código para chamada de relatório de ponto

A Figura 42 apresenta a tela principal enfatizando a forma de criação de um relatório utilizando o ReportBuilder. Um relatório no ReportBuilder é, basicamente, definido por um leiaute e é composto por campos que podem ser parâmetros, variáveis e expressões.



**Figura 42 – Criando um relatório**

Campos denominados *fields* são as áreas que receberam diretamente os dados das colunas selecionadas. Parâmetros são dados passados para operações de preenchimento. Variáveis são utilizadas para armazenar resultados necessários para a geração do relatório. Expressões são utilizadas para especificar o conteúdo de campos de texto ou na realização de cálculos.

Na Listagem 9 é apresentado o código de chamada de um relatório no ReportBuilder. Como o relatório é um componente, é necessário somente passar o nome do componente seguido de `.Print`. No código exposto na Listagem 9, o sistema está carregando os dados que serão impressos no `ClientDataSet cdsRhPonto`. Esse cliente se conectará com um `DataSet`, que se conectará com um `DbPipeline` e este a um componente de relatório. Desta forma todos os dados que forem carregados no `ClientDataSet` estarão disponível ao `DbPipeline` e conseqüentemente no relatório.

```
procedure TFRhRelPonto.DoPrint;
begin
  cdsRhPonto.Close;
  cdsRhPonto.Data := dmConexao.QueryPegaData('SEL_RELPONTO',
      ' ',
      ['P', 'ESTAB', whEstabs.Selecoes,
      'P', 'IDEMPREGADO', whEmpregados.GetAsString,
      'P', 'DINI', whDatas.DataIni,
      'P', 'DFIN', whDatas.DataFin],
      [ftString, ftString, ftTimeStamp, ftTimeStamp],
      [10000, 1000, 0, 0]);

  if (cdsRhPonto.IsEmpty) then
    BibErr.CriaErro('Nenhum Registro Encontrado');

  if (EB_ORDEM.ItemIndex = 0) then
    cdsRhPonto.IndexFieldNames := 'ESTAB;IDEMPREGADO;DATA'
  else
    cdsRhPonto.IndexFieldNames := 'ESTAB;NOME;DATA';
  Report.Print;
end;
```

Listagem 9 - Código de chamada de relatório de ponto

## 5 CONCLUSÃO

O uso de conceitos da orientação a objetos permite representar um sistema por meio de unidades ou elementos mais próximos à sua definição real, isto é os objetos de negócio podem ser mais facilmente representados como unidades computacionais. A modelagem possibilitou a representação do sistema como um conjunto de objetos que se relacionam e assim foi possível verificar se esses objetos representavam a solução planejada para o problema. Esse problema era a leitura dos dados do equipamento de registro de ponto e a sua importação para o sistema de folha de pagamento.

Verificou-se, assim que a utilização de modelos é fundamental para auxiliar na modelagem e na implementação de um sistema tendo como base a orientação a objetos. Os modelos permitem uma prévia do produto final, bem como a sua estrutura e comportamento. No desenvolvimento de sistemas, eles servem como um guia de construção.

Por meio da modelagem pode-se elaborar, de maneira teórica e gráfica, a documentação para todas as etapas do *software* desenvolvido e prever possíveis mudanças ou falhas na estruturação do *software*. Além disso, a ferramenta JUDE Community possibilitou uma visualização da proposta de solução para o sistema.

O sistema desenvolvido alcançou os objetivos propostos. Os dados podem ser importando do equipamento que faz o registro de ponto e enviados para o sistema de folha de pagamento. A definição do leiaute do arquivo de registro de ponto fornece maleabilidade ao sistema porque é possível definir a formatação desses dados importados.

## REFERÊNCIAS

BOOCH, G.; RUMBAUGH, J.; JACOBSON, I.. **UML guia do usuário**. 7ª edição, Rio de Janeiro: Campus, 2000.

CAMPOS, A. **A IDE para modelagem de dados JUDE**. Disponível em: <<http://br-linux.org/linux/node/333>>. Acesso em: 25 out. 2008.

CANTÙ, M. **Dominando o Delphi 7 “A Bíblia”**. São Paulo: Makron Books, 2003.

CANTU, C. H. **Get to know Firebird in 2 minutes (Conheça o Firebird em dois minutos)**. Disponível em: <<http://www.firebirdnews.org/docs/f>>. Acesso em: 26 out. 2010.

FERREIRA, R. **UML - Linguagem de UML Modelagem Unificada**, e-tecnologia.com, 2009. Disponível em: <[http:// http://www.slideshare.net/Ridlo/uml-1858376](http://http://www.slideshare.net/Ridlo/uml-1858376)>. Acesso em: 20 maio 2011.

FURLAN, J. D. **Modelagem de objetos através da UML**. São Paulo: Makron Books, 1998.

IBEXPERT. **IBExpert developer studio**. Disponível em: <<http://www.ibexpert.com/>>. Acesso em: 2 abr. 2011.

MARTIN, J. **Análise e projeto orientados a objeto**. São Paulo: Makron Books, 1995.

PRESSMAN, R. **Engenharia de software**. Rio de Janeiro: Mc Graw-Hill, 2005.

RICARTE, I. L. M. **Introdução a orientação a objetos**. Disponível em: <[http://www.dca.fee.unicamp.br/cursos/POO\\_CPP/node3.html](http://www.dca.fee.unicamp.br/cursos/POO_CPP/node3.html)>. Acesso em: 16 fev. de 2011.

RUMBAUGH, J.; BLAHA, M.; PREMERLANI, W.; EDDY, F.; LORENSEN, W. **Modelagem e projeto baseado em objeto**. Rio de Janeiro: Campus, 1997.

SIEDLER, M. S. **Orientação a objetos**. Disponível em: <[187.7.106.14/marcelo/linguagemwebII/3\\_OrientacaoObjetos97.ppt](http://187.7.106.14/marcelo/linguagemwebII/3_OrientacaoObjetos97.ppt)>. Acesso em: 12 maio 2011.

UML 2. 0. **Introduction to OMG's unified modeling language™ (UML®)**. Disponível em: <[http://www.omg.org/gettingstarted/what\\_is\\_uml.htm](http://www.omg.org/gettingstarted/what_is_uml.htm)>. Acesso em: 20 jul. 2011.