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

MARCIO PADILHA SIQUEIRA

SISTEMA PARA GERENCIAMENTO DE TERCEIRIZAÇÃO DE IMPRESSÃO

PATO BRANCO
2011

MARCIO PADILHA SIQUEIRA

SISTEMA PARA GERENCIAMENTO DE TERCEIRIZAÇÃO DE IMPRESSÃO

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 T. Borsoi

PATO BRANCO
2011

ATA Nº: **176**

DEFESA PÚBLICA DO TRABALHO DE DIPLOMAÇÃO DO ALUNO **MáRCIO PADILHA SIQUEIRA**.

Às 20:00 hrs do dia 1 de julho de 2011, Bloco S da UTFPR, Campus Pato Branco, reuniu-se a banca avaliadora composta pelos professores Beatriz Terezinha Borsoi (Orientadora), Géri Natalino Dutra (Convidado) e Leo João Gava (Convidado), para avaliar o Trabalho de Diplomação do aluno Márcio Padilha Siqueira, matrícula 1030760, sob o título **Sistema para Gerenciamento de Terceirização de Impressão**; 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 20:30 hrs foi encerrada a sessão.

Profa. Beatriz Terezinha Borsoi, Dr.
Orientadora

Prof. Géri Natalino Dutra, M.Sc.
Convidado

Prof. Leo João Gava, Esp.
Convidado

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

Prof. Edilson Pontarolo, Dr.
Coordenador do Curso

RESUMO

SIQUEIRA. Marcio Padilha. Sistema para gerenciamento de terceirização de impressão. 2011. 46 f. Monografia (graduação de Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas) - Universidade Tecnológica Federal do Paraná. Pato Branco, 2011.

Os sistemas de informação podem ser visto como uma ferramenta de suporte para o gestor e de auxílio na realização das atividades relacionadas a bens e serviços. O uso de orientação a objetos na modelagem desse tipo de sistemas traz facilidades como a possibilidade de visualizar a solução do problema antes de a mesma ser implementada. Isso porque é possível definir as entidades de negócio como objetos e como elas interagem entre si. Na implementação o uso de orientação a objetos provê facilidades como o encapsulamento e o reuso. Como forma de auxiliar no gerenciamento de um segmento de mercado, o da terceirização de impressão, este trabalho tem como objetivo o desenvolvimento de um sistema (software) para controle desse tipo de serviço. O sistema foi desenvolvido utilizando a linguagem Delphi que possibilita o uso de conceitos de orientação a objetos.

Palavras-chave: Sistemas de informação. Linguagem Delphi. Orientação a objetos.

LISTA DE QUADROS

LISTA DE FIGURAS

LISTAGENS DE CÓDIGO

LISTAGEM DE SIGLAS

ACID	Atomicidade, Consistência, Isolamento e Durabilidade
AOO	Análise Orientada a Objetos
HTML	*HyperText Markup Language*
IDE	*Integrated Development Environment*
ODBC	*Open Database Connectivity*
OLEDB	*Object Linking and Embedding, Database*
PDF	*Portable Document Format* (
POO	Projeto Orientado a Objetos
PSQL	*Procedural Structured Query Language*
RTF	*Rich Text Format*
SQL	*Structured Query Language*
TMO	Técnicas de Modelagem de Objetos
UML	*Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

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

1.1 CONSIDERAÇÕES INICIAIS

As tecnologias de informação e comunicação têm facilitado a exposição de produtos e das empresas ao consumidor. Consequentemente, o mercado está cada vez mais competitivo. A competitividade e a facilidade de pesquisar por marcas, requisitos e preços, que é proporcionada pelas tecnologias de informação e comunicação, contribuem para que as empresas estejam em constante busca por soluções melhores para a venda dos seus produtos e da realização dos seus serviços.

O *outsourcing* de impressão é um serviço prestado por um fornecedor externo para otimizar ou gerir todo o serviço de impressão de uma empresa. Normalmente o fornecedor tem a responsabilidade de fornecer todos os recursos para a impressão, incluindo equipamentos, suprimentos e *software* de bilhetagem.

Os provedores desse tipo de serviço estão sempre em busca de novas tecnologias para desenvolver soluções de baixo custo, seja pela concorrência seja pela necessidade de agregar valor à locação de equipamentos. Dentre os principais benefícios para quem contrata serviço de terceirização de impressão estão uma redução de cerca de 40% de despesas com impressão, segundo dados levantados na empresa Net-ON. Além de um controle de tudo o que está envolvido com o serviço de impressão.

Considerando a importância e a oportunidade de desenvolver um sistema que possa auxiliar no controle de prestação de serviços de terceirização de impressão, este trabalho apresenta conceitos de análise orientada a objetos e da UML *(Unified Modeling Language),* aplicando-os na modelagem e na implementação de um sistema para controle administrativo de empresas que prestam esse tipo de serviço. A modelagem orientada a objetos auxilia para que o sistema seja implementado de forma a atender aos requisitos estabelecidos. Isso porque as entidades que compõem o negócio são representadas como objetos que possuem atributos, realizam operações e se comunicam entre si por meio de troca de mensagens.

1.2 OBJETIVOS

O objetivo geral está relacionado ao resultado principal da realização deste trabalho e os objetivos específicos visam agregá-lo.

1.2.1 Objetivo Geral

- Implementar um sistema de controle administrativo para empresas de terceirização de Impressão, aplicando conceitos de análise orientada a objetos.

1.2.2 Objetivos Específicos

- Apresentar conceitos de orientação a objetos e de sua modelagem utilizando a UML.
- Exemplificar o uso da linguagem Delphi na implementação de um sistema para controle de serviço de terceirização de impressão.

1.3 JUSTIFICATIVA

A proposta do sistema computacional obtido como resultado deste trabalho tem como base a necessidade verificada de um controle dos gastos relacionados a suprimentos e material para a locação dos equipamentos de informática, mais especificamente para o segmento de terceirização de serviço de impressão. A empresa que foi utilizada como estudo de caso, a Net-ON, necessita desse controle.

Com o uso de um sistema informatizado as empresas que prestam esse tipo de serviço reduzirão o tempo no processamento das informações de controle que elas necessitam manipular. As planilhas que hoje levam horas para serem preenchidas e atualizadas diariamente, com poucos minutos estarão prontas. Um sistema computacional fornecerá os indicadores e as informações precisas, oportunas e acessíveis, facilitando e proporcionando a realização de um gerenciamento mais eficiente.

A linguagem Delphi foi escolhida como ferramenta de programação, por ser

adequada para o desenvolvimento de soluções corporativas. Além disso, essa linguagem possui um compilador considerado rápido e eficiente para as necessidades do sistema a ser implementado e facilita a implementação orientada a objetos.

A análise orientada a objetos foi utilizada para que fosse possível beneficiar-se do uso da orientação a objetos na linguagem Delphi e por facilitar a modelagem e a implementação. Além de uma melhor produtividade a longo prazo, permitindo a reutilização de código pelos desenvolvedores.

1.4 ORGANIZAÇÕES 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 do trabalho apresentando conceitos de orientação a objetos e sua modelagem utilizando a UML.

No Capítulo 3 está o método, que é a seqüência geral de passos para realizar o trabalho. Os materiais se referem ao que foi utilizado para modelar o sistema, incluindo uma ferramenta de gerenciamento de projeto.

O Capítulo 4 contém o resultado do trabalho que é a modelagem de um sistema para gerenciamento de empresas de terceirização de impressão.

No Capítulo 5 está a conclusão com as considerações finais.

2 FUNDAMENTAÇÃO TEÓRICA

A fundamentação teórica do trabalho tem como base a orientação a objetos, considerando a análise e a modelagem de sistema utilizando a UML.

2.1 ORIENTAÇÃO A OBJETOS

As técnicas orientadas a objeto permitem que o *software* seja construído (modelado e implementado) como um conjunto de objetos e que cada objeto possui dados e comportamento específicos. Embora, os próprios objetos possam ser construídos a partir de outros objetos.

Essas técnicas se preocupam, principalmente, com o comportamento do objeto. A estruturação dos elementos do programa pode ser especificada e documentada utilizando a UML. A UML é uma linguagem padrão desenvolvido com o propósito de visualizar os resultados da análise de sistemas (BOOCH, RUMBAUGH, JACOBSON, 1999).

A estrutura básica da modelagem - análise e projeto - baseados em objetos é o objeto. Um objeto combina a sua estrutura composta por dados e o comportamento desses dados em uma única entidade (BOOCH, RUMBAUGH, JACOBSON, 1999). Técnicas de modelagem de objetos abrangem desde a definição inicial do sistema até a sua implementação. Assim, objetos são utilizados em contextos e de formas distintas.

Os objetos do domínio da aplicação compõem a estrutura do modelo em termos de negócio e não de objetos computacionais. Desse modelo são definidas as classes com os atributos e as operações que serão realizadas pelos respectivos objetos. Isso ocorre na fase de análise e de projeto. Esses modelos são utilizados na definição das tabelas com seus campos que compõem o banco de dados e das classes utilizadas na implementação, com seus atributos e métodos. As classes podem ser definidas considerando mecanismos como herança e os métodos por polimorfismo.

O projeto de sistemas orientados a objetos inicia com a análise do domínio da aplicação e termina somente quando o sistema estiver totalmente implementado. Os conceitos identificados a partir da abstração dos requisitos da aplicação são mapeados em classes que se comunicam por um mecanismo denominado troca de

mensagens.

Uma metodologia baseada em objetos consiste na construção de um modelo de um domínio da aplicação e na posterior adição a este dos detalhes de implementação durante o projeto do sistema. Essa abordagem referenciada como TMO (Técnicas de Modelagem de Objetos) por Rumbaugh et al. (1997), é composta das seguintes fases (POTOK, MLADEN, 1995):

a) Análise – o modelo de análise é uma abstração concisa e precisa do que o sistema deverá fazer. Os objetos desse modelo devem ser conceitos do domínio da aplicação e não conceitos de implementação computacional, como, por exemplo, a estrutura de dados. A Análise Orientada a Objetos (AOO) é o processo que define um modelo que resolve um determinado problema ou encontra um conjunto de requisitos. Esse modelo é, em termos do mundo real, o espaço do problema. Consiste de um modelo de objetos com as interações entre os mesmos e pode incluir modelo de processo ou estados e análise de requisitos.

b) Projeto do sistema e dos objetos – o projeto é a organização do sistema em subsistemas tendo como base as classes e a arquitetura proposta. O projeto dos objetos constrói um modelo baseado na análise, mas contendo detalhes de implementação. O enfoque desse projeto são as estruturas de dados e os algoritmos necessários à implementação de cada classe. As classes de objetos provenientes da análise são significativas, mas são acrescidas das estruturas de dados do domínio computacional e de algoritmos elaborados para resolver, implementar, a solução definida.

Na fase de Projeto Orientado a Objetos (POO) ocorre a transformação do modelo de análise na descrição de projeto. Essa fase inclui decomposição dos objetos em nível de análise, avaliação para utilização de componentes existentes e iterativamente criar uma hierarquia de objetos. A decomposição do modelo de análise é idealmente um passo de refinamento, produzindo um mapeamento entre o modelo de análise e o sistema final.

c) Implementação – na implementação as classes e os relacionamentos são traduzidos, por meio de linguagens de programação, no código que compõem o programa. A implementação orientada a objetos é um refinamento do modelo de projeto em um *software* executável. O uso da orientação a objetos pode aumentar a produtividade porque a AOO e o POO, se feitos corretamente, facilitam a implementação, os testes e provem o reuso, além de reduzir o esforço de

manutenção.

Dentre os conceitos relacionados à orientação a objetos estão: objeto, atributo, método, classe, mensagem, encapsulamento, polimorfismo e herança. Esses conceitos são apresentados a seguir.

a) Objeto

Para Martin (1995, p. 18) “um objeto é qualquer coisa, real ou abstrata, a respeito da qual armazenamos dados e os métodos que os manipulam”.

Em uma aplicação computacional, um objeto está relacionado à abstração de algo que possui fronteira definida e significado para a aplicação. Um objeto é capaz de armazenar estados por meio de seus atributos e reagir a mensagens enviadas a ele, assim como se relacionar e enviar mensagens a outros objetos.

Por exemplo, uma impressora com os atributos marca e número de série define uma classe. E uma impressora da marca ABC que possui o número de série 345543245-23 é um objeto dessa classe impressora.

b) Atributo

Os atributos correspondem aos elementos da classe que armazenam as características dos objetos instanciados. É o valor de dado armazenado pelos objetos de uma classe. Cada atributo possui um valor para uma instância de objeto. Na definição de uma classe, o nome do atributo pode ser seguido por dados opcionais como o tipo e o valor padrão.

c) Método

Os métodos especificam a maneira pela qual os dados de um objeto são manipulados. Os métodos de um tipo de objeto referenciam somente as estruturas de dados desse tipo de objeto. Eles não devem acessar diretamente as estruturas de dados de outro objeto. Para usar a estrutura de dados de outro objeto, eles devem enviar uma mensagem a esse objeto. O tipo de objeto empacota, num mesmo bloco, os tipos de dados e os métodos.

d) Mensagem

Uma mensagem é uma solicitação para executar a operação indicada sobre determinados atributos do objeto e retornar o resultado (MARTIN, 1995). Consequentemente, as implementações orientadas a objetos referem-se às mensagens como solicitação.

Para fazer com que um objeto execute algo é preciso enviar uma solicitação. Esta solicitação fará com que a operação seja executada. As mensagens são

constituídas pelo nome do objeto, o nome da operação e, eventualmente, um grupo de parâmetros.

e) Classes

Martin (1995) define que uma classe de objetos descreve um grupo de objetos com propriedades semelhantes (atributos), o mesmo comportamento (operações), os mesmos relacionamentos com outros objetos e a mesma semântica.

Normalmente os objetos derivam a sua individualidade da diferença de seus valores e atributos e relacionamento com outros objetos, mas é possível a existência de objetos com valores de atributos e relacionamentos idênticos.

Para Silva Filho (2011), as classes não são suportadas em todas as linguagens, porém são absolutamente necessárias às linguagens que sejam orientadas a objetos. A maioria das linguagens de programação baseadas em objeto podem determinar a classe de um objeto em tempo de execução, a classe é uma propriedade implícita no objeto.

f) Encapsulamento

O ato de empacotar ao mesmo tempo dados e métodos é denominado encapsulamento. O objeto esconde seus dados de outros objetos e permite que os dados sejam acessados por meio de seus próprios métodos. O encapsulamento protege os dados de um objeto de acesso ou manipulação indevida e oculta os detalhes de implementação interna do objeto aos seus usuários.

O encapsulamento é importante porque separa a maneira como um objeto se comporta da maneira como ele é implementado. Isso permite que as implementações do objeto sejam modificadas sem exigir que os aplicativos que as usam sejam também modificados.

É mais fácil de modificar programas usando-se o encapsulamento porque um tipo de objeto é modificado de uma vez só. Se um tipo de objeto for modificado, somente os métodos e as estruturas de dados associados a esse tipo de objeto são afetados, e usualmente apenas alguns desses métodos e estruturas de dados o são. O comportamento do tipo de objeto pode ser modificado e testado, independentemente de outros tipos de objeto.

g) Polimorfismo

Polimorfismo significa que uma mesma operação pode assumir muitas formas de implementação, dependendo do tipo de objeto. Com polimorfismo uma solicitação para uma operação pode ser feita sem saber qual método deve ser invocado.

h) Herança

Herança é o processo que permite o compartilhamento de atributos e operações entre classes baseada em uma hierarquia. Martin (1995, p. 26) explica que “uma subclasse herda as propriedades da sua classe-mãe; uma subclasse herda as propriedades das subclasses e assim por diante”. 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*.

Assim, herança é a capacidade de um novo objeto tomar atributos e operações de um objeto existente, ou seja, permite criar classes complexas sem repetir código. A nova classe herda as características da classe base. E quando a nova classe herda o comportamento e os atributos de mais de uma classe, é denominada herança múltipla.

2.2 UML

A UML é uma linguagem padrão para a elaboração da estrutura de projetos de *software*. A UML é uma linguagem de diagramas para documentar, especificar, construir e visualizar modelos de *software* orientados por objetos.

A UML é composta por muitos elementos de modelo que representam as diferentes partes de um sistema de *software.* Os elementos UML são usados para criar diagramas, que representam uma determinada parte, ou um ponto de vista do sistema (FURLAN, 1998).

2.2.1 Diagramas da UML

Um sistema dificilmente pode ser compreendido em sua totalidade e pelos diversos envolvidos baseando-se apenas em única representação. A UML define diversos diagramas que possibilitam representar o sistema sob perspectivas ou interesses distintos (FURLAN, 1998).

A Figura 1 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.

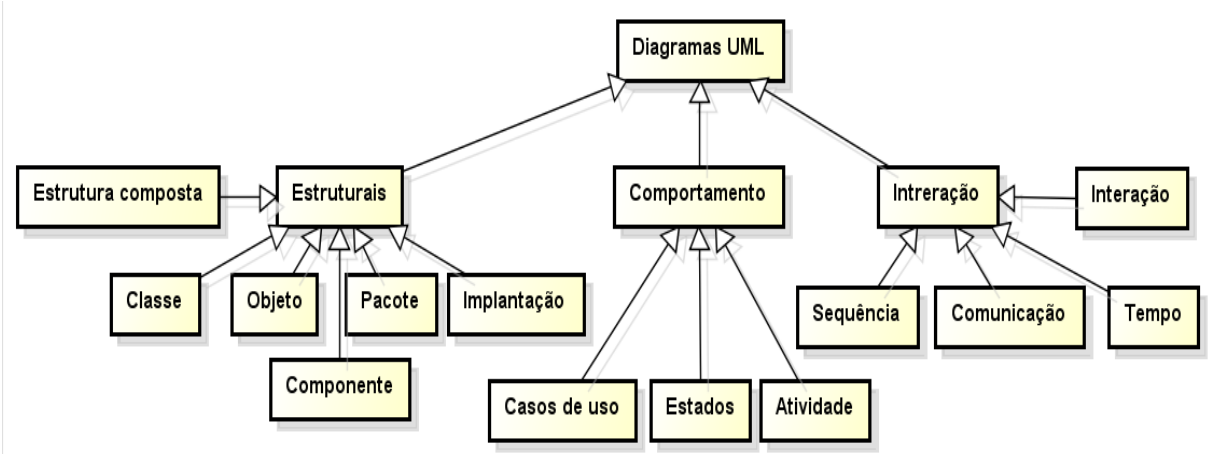


Figura 1 – Diagramas da UML versão 2
Fonte: composto com base em informações constantes em UML 2.0 (2011)

A seguir são apresentados os diagramas de casos de uso e de classes por serem os tipos utilizados na realização deste trabalho.

a) Diagrama de casos de uso

É um diagrama usado para identificar como o sistema se comporta em várias situações que podem ocorrer durante sua operação. Descreve o sistema, seu ambiente e a relação entre os dois. Os componentes deste diagrama são os atores e os casos de uso.

Ator representa qualquer entidade que interage com o sistema. Pode ser uma pessoa, um dispositivo fisco ou outro sistema. Algumas das características atribuídas aos atores são:

- Não é parte do sistema. Representa os papéis que o usuário do sistema pode desempenhar;
- Pode interagir ativamente com o sistema;
- Pode ser um receptor passivo de informação.

Caso de uso é uma sequência de ações que o sistema executa e produz um resultado de valor para o ator. Algumas de suas características são:

- Um caso de uso modela o diálogo entre atores e o sistema.
- Um caso de uso é utilizado por um ator para invocar certa funcionalidade do sistema.
- Um caso de uso representa um fluxo de eventos completo e consistente.
- O conjunto total dos casos de uso representa as situações possíveis de

utilização do sistema.

A Figura 2 representa um exemplo genérico de diagrama de caso de uso, no qual o ator interage com funções providas pelo sistema.

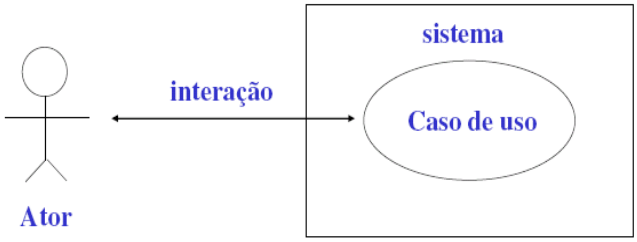


Figura 1 - Diagramas de caso de uso

As interações entre os atores e os casos de uso são bilaterais, no sentido que um ator pode iniciar um caso de uso e que o caso de uso pode informar o ator de ações realizadas.

b) Diagrama de classes

Um diagrama de classes define as classes que compõe um sistema, sua estrutura com informações sobre métodos, atributos, nome das funções e como serão relacionadas. As relações entre as classes podem ser agregação, herança ou associação. Uma relação indica um tipo de dependência entre classes. Essa dependência pode ser definida como forte ou fraca e indicam que as classes relacionadas cooperam de alguma forma para cumprir um objetivo para o sistema.

A Figura 3 representa no retângulo mais à esquerda a representação esquemática de uma classe, no retângulo central o exemplo de uma classe (impressora) e no retângulo mais à direita um exemplo de objeto. Esse objeto é uma instanciação da classe impressora.

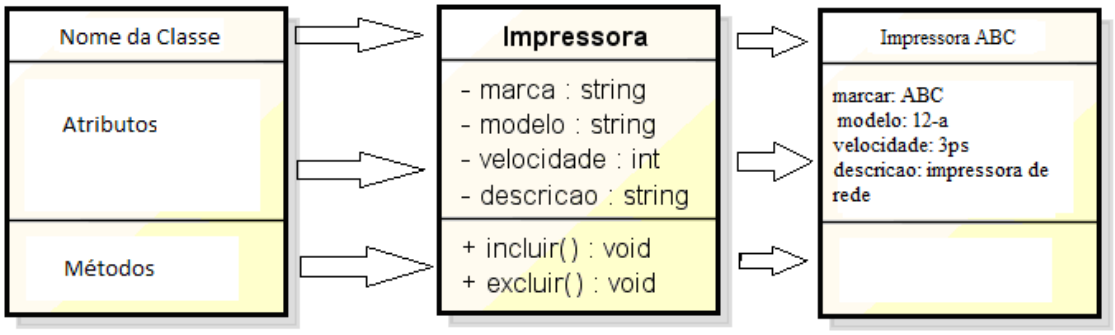


Figura 2 - Exemplo de uma classe

3 MATERIAIS E MÉTODO

Este capítulo apresenta os materiais utilizados na realização do trabalho e o método empregado para a condução das atividades.

3.1 FERRAMENTAS E TECNOLOGIAS

A ferramenta Astah Community foi atualizada para a construção dos diagramas. A escolha da ferramenta Astah deu-se devido a sua facilidade de interação com o usuário e por ser gratuita. Para a implementação do sistema foi utilizada a linguagem Delphi com a sua própria interface de desenvolvimento, com o banco de dados Firebird e Report Builder para a confecção dos relatórios.

3.1.1 Astah Community

A ferramenta Astah Community é empregada para realizar a modelagem de projetos utilizando a UML. A Figura 4 apresenta a tela principal dessa ferramenta.

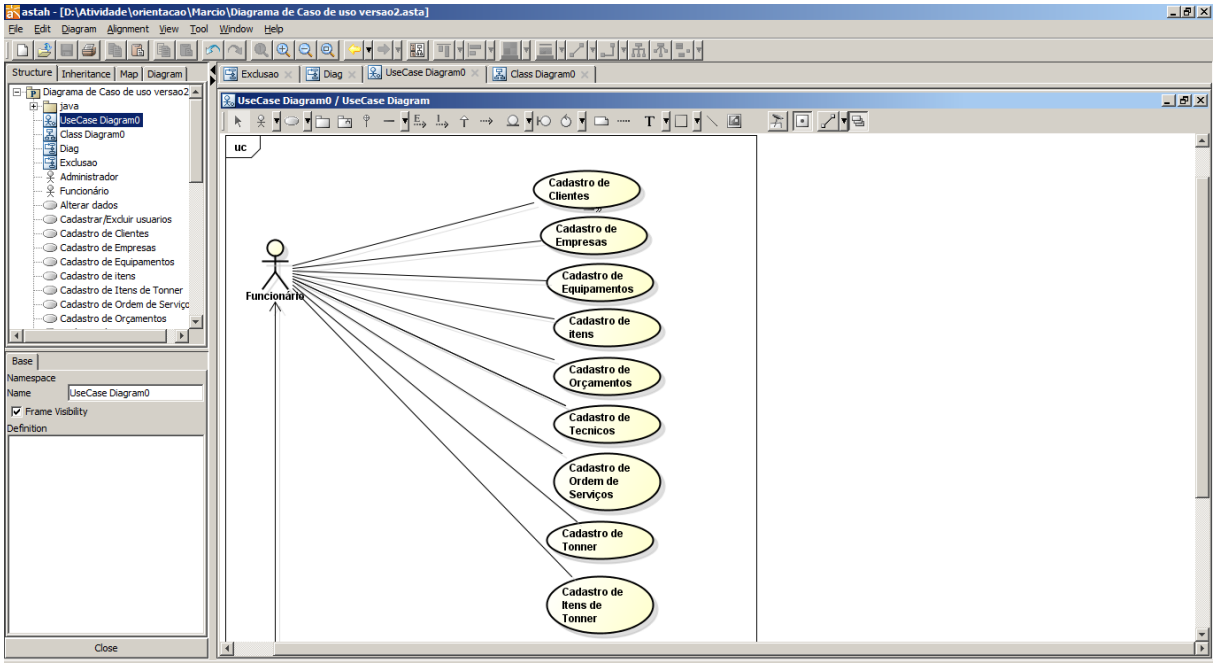


Figura 4 - Tela principal do Astah Community

A ferramenta Astah está organizada em três partes básicas:

a) Organização do projeto (área superior à esquerda da Figura 4) – essa área possui abas com visões diferentes do projeto, são elas: *Structure* (árvore de

estrutura do projeto), *Inheritance Tree* (exibe as heranças identificadas), *Map* (exibe todo o editor de diagrama) e *Diagram* (mostra a lista de diagramas do projeto).

b) Visão das propriedades (área inferior à esquerda central da Figura 4) – nessa área, as propriedades dos elementos dos diagramas são apresentadas para edição. Por exemplo, com um diagrama de classes aberto e uma classe selecionada, são exibidas todas as propriedades da classe, como: nome, visibilidade, atributos e operações.

c) Editor do diagrama (área direita da Figura 4) – é a área na qual são exibidos e editados os diagramas. Com um duplo clique em um determinado diagrama exibido na lista de diagramas, o mesmo é carregado nesta área e todos os seus elementos são mostrados. Os diagramas são compostos por meio dos componentes expostos na barra que está na parte superior dessa área. Essa barra apresenta os componentes de acordo com o tipo de diagrama que é selecionado pelo menu *Diagram*.

3.1.2 Delphi 7

O ambiente de desenvolvimento Delphi se baseia em uma extensão orientada a objetos da linguagem de programação Pascal, também conhecida como Object Pascal (CANTU, 2003). Constitui-se em um ambiente de desenvolvimento por ser composto de diversas ferramentas agregadas.

Como ambiente de desenvolvimento, Delphi é um compilador e também uma IDE (*Integrated Development Environment*) para o desenvolvimento de *software*. A linguagem utilizada pelo Delphi, o *Object Pascal* (Pascal com extensões orientadas a objetos), a partir da versão 7.0 passou a se chamar Delphi Language. Delphi é um IDE que foi produzida pela Borland Software Corporation para a linguagem de programação Delphi.

Dentre as características da linguagem Delphi, destacam-se (CANTU, 2000; CANTU, 2003):

b) RAD (*Rapid Application Development*), agiliza o desenvolvimento de aplicações.
c) Construtor visual de interface com o usuário, formulário e componentes.
d) Arquitetura baseada em componentes (reutilização e manutenção).
e) Compilador de código nativo.

f) O código é gerado automaticamente durante a montagem do formulário.

Biblioteca de Componentes Visuais, incluindo objetos padrão de interface com o usuário, gerenciamento de dados, gráficos e multimídia, gerenciamento de arquivos.

i) Arquitetura aberta. Possibilidade de adicionar componentes e ferramentas personalizadas de terceiros.

j) Project Manager: oferece uma visualização de todos os *forms* e *units* de um determinado projeto e oferece um mecanismo conveniente para gerenciar projetos

A Figura 5 apresenta a tela inicial de IDE da linguagem Delphi.

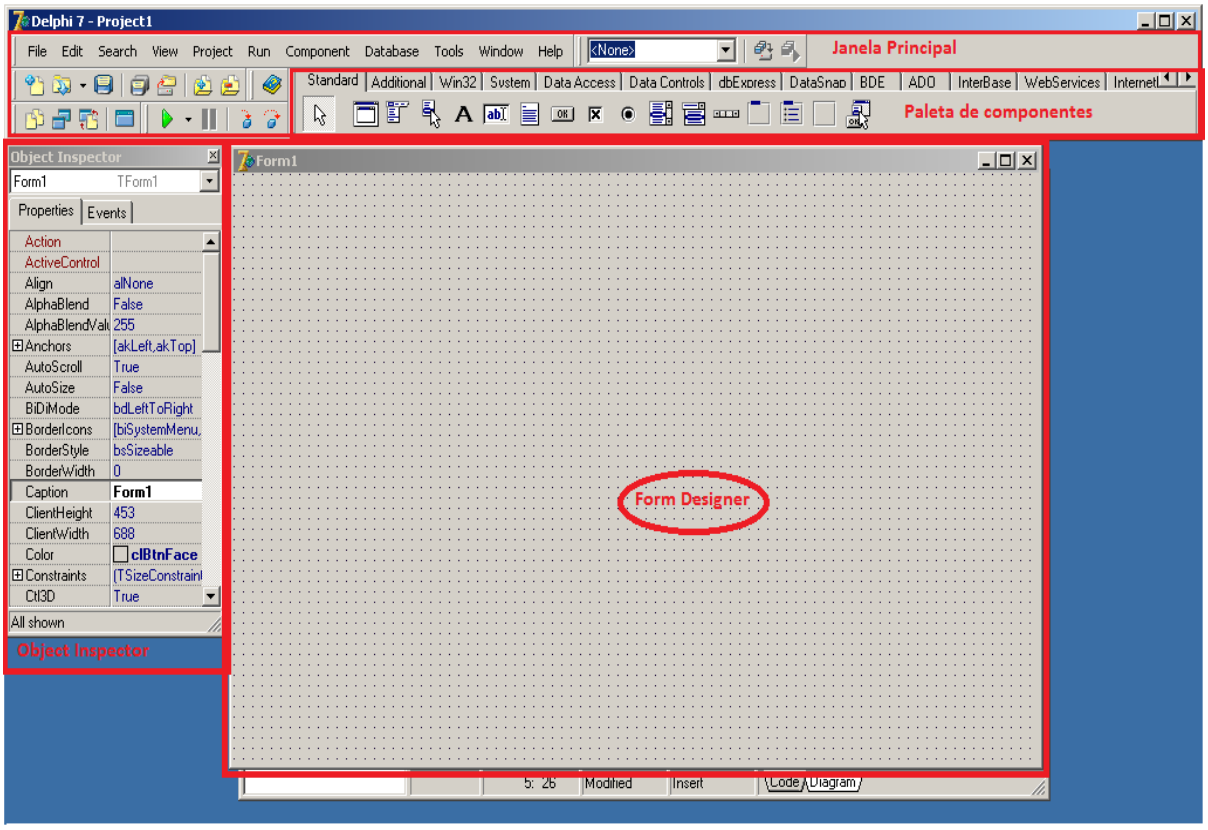


Figura 5 - Tela inicial de IDE da linguagem Delphi

As partes marcadas na Figura 5 indicam:

a) Paleta de componentes – a paleta de componentes é uma barra de ferramentas com diversos componentes para serem utilizados no desenvolvimento de aplicações. A paleta é constituída de várias guias. Cada guia possui vários componentes agrupados de acordo com sua finalidade.

b) *Form designer* - o *form* é uma janela para desenvolvimento da interface gráfica da aplicação. É no *form* que os componentes são colocados para compor a camada de interação da aplicação.

c) *Object inspector* - define propriedades e eventos para os componentes. Propriedades são características do componente selecionado. Eventos são suportados pelos respectivos componentes. O *Object inspector* muda de acordo com o componente escolhido.

3.1.3 Firebird

O banco de dados Firebird é derivado do código do Borland InterBase 6.0. Ele tem o código fonte aberto e gratuito. O Firebird é um SGBD completo e possui suporte nativo para diversos sistemas operacionais - incluindo Windows, Linux, Solaris e MacOS - *triggers* de conexão e transação (CANTU, 2010).

Dentre os recursos do Firebird destacam-se: transações compatíveis com ACID (Atomicidade, Consistência, Isolamento e Durabilidade); integridade referencial; utiliza poucos recursos de processamento; linguagem nativa para *stored procedures* e *triggers* (PSQL - *Procedural Structured Query Language*); suporte para funções externas; versão *embedded* do SGBD; diversas formas de acesso ao banco de dados, como: nativo/API, dbExpress, ODBC (*Open Database Connectivity)*, OLEDB (*Object Linking and Embedding, Database*), .Net provider, JDBC nativo tipo 4, Python module, PHP, Perl; *backups* incrementais; e tabelas temporárias.

3.1.4 Sistema Gerenciador de Banco de Dados IBExpert

IBExpert (IBEXPERT, 2011) é uma ferramenta para administração de bancos de dados Interbase e Firebird. Permite criar e gerenciar usuários e tabelas. Para utilizar o gerenciador é necessário registrar o banco de dados para que o mesmo seja reconhecido. A interface do aplicativo é bastante simplificada consistindo basicamente de uma barra de ferramentas e do DBExplorer.

A ferramenta IBExpert foi escolhida por ser gratuita e de fácil utilização e pode ser baixada gratuitamente a partir do site oficial da ferramenta (www.ibexpert.com). IBExpert permite criar e gerenciar usuários e tabelas. Para utilizar o gerenciador é necessário registrar o banco de dados. Possui alguns recursos interessantes, como o alimentador automático de tabelas que insere dados de teste nas tabelas.

A Figura 6 apresenta a tela inicial da ferramenta IBExpert.

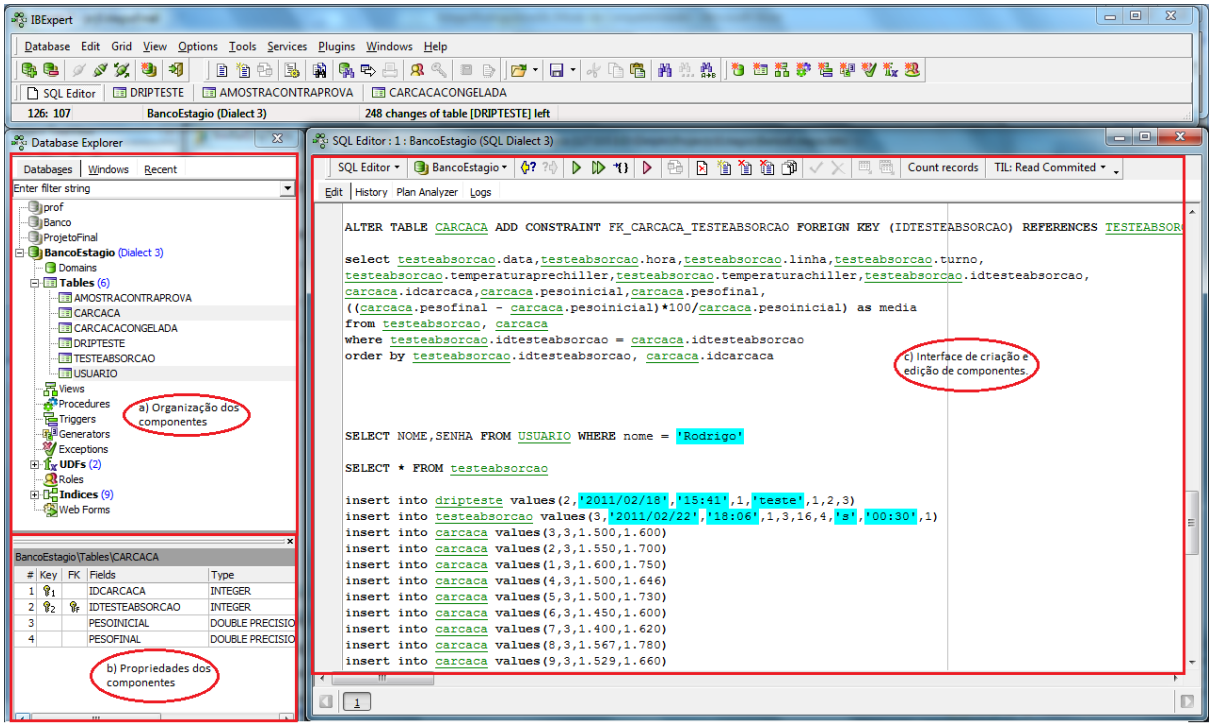


Figura 6 - Tela inicial do sistema gerenciador de banco de dados IBExpert

O sistema gerenciador de banco de dados IBExpert está organizado em três partes básicas, destacadas na Figura 6:

a) Organização dos componentes - nesta parte da interface do IBExpert o usuário tem uma visão dos bancos de dados que foram criados, também pode criar uma nova base de dados, além de poder fazer uma série de alteração dos banco de dados já existentes, bem como criar tabelas de forma visual.

b) Propriedades dos componentes - fornece para o usuário uma visão das propriedades do componente selecionado na tela de organização dos componentes.

c) Interface de criação e edição de componentes - para a definição das tabelas por meio de linha de código (SQL - *Structured Query Language*), bem como inserir e editar dados, selecionar visualização das tabelas e dos dados que estão armazenados nas tabelas.

3.1.5 Report Builder

O Report Builder é uma ferramenta de geração de relatórios para plataformas Linux e Windows. As principais características e funcionalidades do Report Builder são (REPORT BUILDER, 2011):

- Possui um editor visual para criação de relatórios integrado ao ambiente;
- Os relatórios podem ser salvos nos formatos RTF (*Rich Text Format*), HTML (*HyperText Markup Language*), PDF (*Portable Document Format*), texto e um formato proprietário (RTM);
- Possui diversos recursos para formatação dos relatórios, como alinhamento e posicionamento de objetos;
- Os relatórios são baseados em dados, cálculo, desenho e visualização de impressão. É possível visualizar e testar os relatórios em tempo de projeto a partir do visualizar impressão;
- Disponibiliza a criação de páginas padrão para serem usadas como base para vários relatórios;
- Possibilita acesso aos objetos do relatório a partir da aplicação Delphi;
- Possui componentes para suporte a código de barras, para codificação de eventos do relatório;

O Report Builder trabalha com o conceito de projetos. Um projeto pode conter vários relatórios. Os relatórios podem ser distribuídos separadamente do executável da aplicação ou incorporados ao mesmo. A Figura 7 apresenta a tela principal do Report Builder.

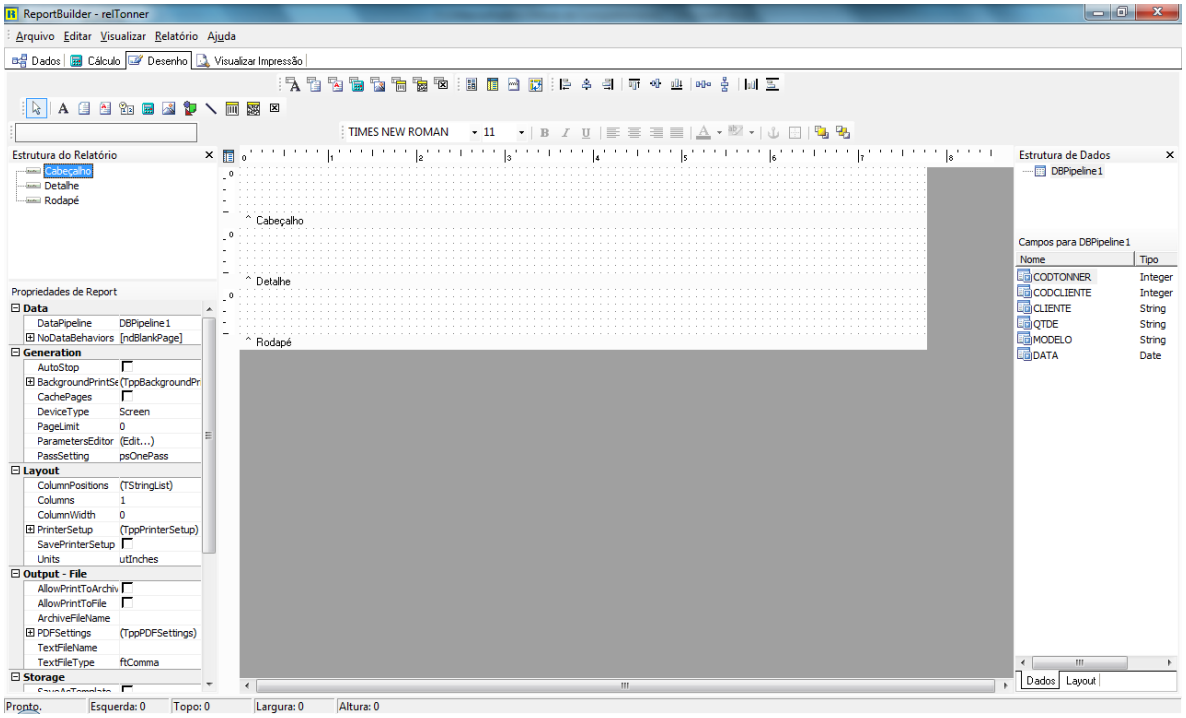


Figura 7 - Tela inicial do Report Builder

3.2 ATIVIDADES REALIZADAS NA IMPLEMNTAÇÃO DO SISTEMA

O sistema foi desenvolvido tendo como base uma empresa específica. O método está centrado no levantamento bibliográfico para o entendimento e uso de conceitos de orientação a objetos, análise e projeto de sistemas e em um estudo de caso. O estudo de caso se refere ao sistema que foi desenvolvido para uma empresa específica, a Net-ON.

A Net-ON, em 2003 abriu um segmento de negócio, especializando em *outsourcing* ou terceirização de equipamentos e serviços. A terceirização é a forma de reduzir os custos e garantir qualidade e disponibilidade no processo de impressão, seja empresa de pequeno ou de grande porte. Terceirizar é fornecer um serviço completo que abrange equipamentos, insumos, peças, serviços, logística e gerenciamento dos equipamentos.

As etapas para a realização deste trabalho estão organizadas em planejamento, levantamento bibliográfico, levantamento de requisitos do sistema e análise e projeto do sistema. Essas etapas estão descritas a seguir.

a) Planejamento

No planejamento foi realizada a organização do trabalho. O sistema teria como base a modelagem desenvolvida para o estágio. Essa modelagem seria melhorada e a implementação realizada. O referencial teórico foi revisto. No planejamento foi definida, ainda, a organização da implementação. Isso no sentido de quais funcionalidades seriam implementadas. Optou-se iniciar pela implementação dos cadastros básicos para que o autor deste trabalho pudesse familiarizar-se com a linguagem de programação.

b) Revisão dos requisitos definidos

Os requisitos definidos como trabalho de estágio foram revistos. Para definição dos requisitos do sistema foram considerados os procedimentos realizados pela empresa para terceirização de impressão, as planilhas de controle de cada tipo de serviços utilizadas, os tipos de equipamentos que a empresa faz locação.

c) Análise de requisitos e projeto do sistema

Os requisitos levantados foram modelados por meio de casos de uso, um diagrama que apresentasse a visão geral do sistema, diagrama de classes e de atividades. O diagrama de classes foi revisto e o banco de dados foi definido. Uma listagem de tabelas e seus respectivos campos foi elaborada.

d) Implementação do sistema

A implementação do sistema ocorreu por meio da linguagem Delphi, com os relatórios definidos utilizando o Report Builder.

e) Testes

Os testes foram realizados pelo autor deste trabalho visando encontrar erros e por funcionários da empresa para verificar se os requisitos foram atendidos. O sistema está em implantação na empresa que serviu de base (estudo de caso) para a definição dos requisitos.

4 SISTEMA PARA GERENCIAMENTO DE TERCEIRZAÇÃO DE IMPRESSÃO

Neste capítulo é apresentado o sistema desenvolvido para automatizar a locação de equipamentos relacionados à terceirização de impressão..

4.1 APRESENTAÇÃO DO SISTEMA

O sistema visa prover o controle de usuários, de equipamentos, de peças e suprimentos utilizados nos clientes com locação. E, ainda, um controle de cópias impressas por clientes, na opção de locação de impressoras.

4.2 MODELAGEM DO SISTEMA

A Figura 8 apresenta o diagrama de casos de uso.

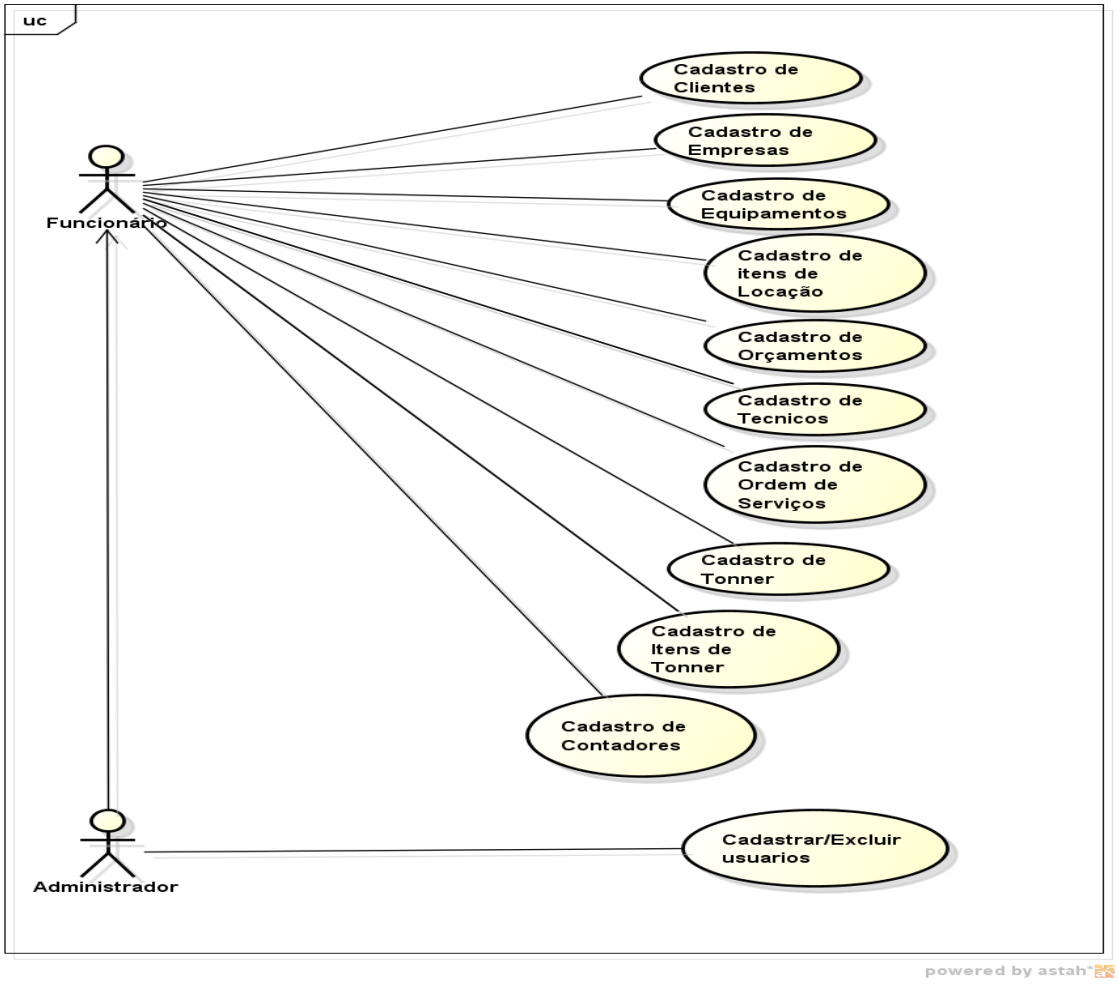


Figura 8 - Diagrama de casos de uso

Os casos de uso constantes na Figura 8 estão descritos no Quadro 1

Caso de uso	Descrição
Alterar dados pessoais	Alteração dos dados pessoais de cadastro no sistema.
Cadastro de clientes	O sistema deve registrar os clientes, cadastrando informações diversas sobre eles. Inclusão – cadastrar clientes no sistema. Exclusão – excluir clientes do sistema. Alteração – alterar algum dado do cliente no sistema. Relatório – listagem dos clientes cadastrados no sistema.
Cadastro de Empresas	O sistema deve registrar as empresas (matriz/filial) no sistema. Inclusão – cadastrar empresas no sistema. Exclusão – excluir empresas do sistema. Alteração – alterar dados de empresas cadastradas. Relatório – listagem das empresas cadastradas.
Cadastro de Equipamentos	O sistema deve registrar os equipamentos disponíveis para locação, este inclui, impressoras, computador, *notebook*, projetor e telão. Inclusão – cadastrar equipamentos no sistema. Exclusão – excluir equipamentos do sistema. Alteração – alterar dados de equipamentos cadastrados. Relatório – listagem dos equipamentos cadastrados no sistema.
Cadastro de itens de locação	O sistema deve registrar os itens usado na locação de equipamentos, incluindo cilíndro, fusor e peças usadas nas impressoras e computadores. Inclusão – cadastrar itens de locação no sistema. Exclusão – excluir itens do sistema. Alteração – Alterar dados de itens cadastrados. Relatório – listagem dos itens cadastrados no sistema.
Cadastro de orçamentos	O sistema deve registrar os orçamentos realizados para os clientes. Inclusão – cadastrar orçamentos no sistema. Exclusão – excluir orçamentos do sistema. Alteração – alterar dados de orçamentos cadastrados. Relatório – listagem dos orçamentos cadastrados no sistema.
Cadastro de técnicos	O sistema deverá registrar os técnicos da empresa para poder utilizar o sistema. Inclusão – cadastrar técnicos no sistema. Exclusão – excluir técnicos do sistema. Alteração – alterar dados de técnicos cadastrados. Relatório – listagem dos técnicos cadastrados no sistema.
Cadastro de ordem de serviço	O sistema deverá registrar as ordens de serviços gerada pela locação de equipamentos e serviços. Inclusão – cadastrar ordens de serviço no sistema. Exclusão – excluir ordens de serviço do sistema. Alteração – alterar dados de ordens de serviço cadastradas. Relatório – listagem das ordens de serviço cadastradas no sistema.
Cadastro de toner	O sistema deverá registrar os tonner que são utilizados nos clientes. Inclusão – cadastrar toner no sistema. Exclusão – excluir toner do sistema. Alteração – alterar dados de toner cadastrados.

	Relatório – listagem dos toner cadastrados no sistema.
Cadastro de Contadores	O sistema deverá registrar contadores de cada cliente através de sua identificação. Inclusão – incluir dado de contador no sistema. Exclusão – excluir dados de contadores cadastrados. Alteração – alterar dados de contadores cadastrados. Relatório – listagem de contadores cadastrados no sistema.

Quadro 1 – Casos de uso do sistema

A seguir requisitos funcionais e não-funcionais detalhados dos principais itens do sistema. O modelo dos quadros que documentam os requisitos forma elaborados de acordo com o constante em Wazlawick (2004). O Quadro 2 contém o requisito funcional Cadastro de Orçamentos. Esse requisito funcional possibilitará ao administrador cadastrar os orçamentos da empresa. Adicional ao requisito funcional está à lista de requisitos não-funcionais associados.

F1 Cadastrar Orçamentos.	**Oculto ( )**			
Descrição: O sistema deve possuir uma área de acompanhamento de orçamentos, provendo a inclusão de novos orçamentos, a lista dos orçamentos e busca por data, por cliente e por tipo de equipamento.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF1.1 Identificação de orçamento.	O sistema deve identificar o orçamento feito pelo cliente através de seu nome.	Segurança	( )	(X)
NF1.2 Identificação da operação	O sistema deve identificar se o orçamento é de Locação de equipamento, de venda ou serviços.	Segurança	( )	(X)
NF1.3 Janela Única	Todas as funções relacionadas aos orçamentos devem ser efetuadas em uma única janela.	Interface	(X)	( )

Quadro 2 – Requisitos de cadastro de orçamentos

O Quadro 3 apresenta o requisito funcional Cadastro de Equipamentos. Esse requisito funcional possibilitará ao administrador ao cadastrar os equipamentos da empresa. Adicional ao requisito funcional está à lista de requisitos não-funcionais associados.

F2 Cadastrar Equipamentos	**Oculto ( )**			
Descrição: O cadastro de equipamentos deve incluir o equipamento para o cliente, uma lista dos equipamentos, busca por nome, CNPJ cliente, tipo de equipamento, por código, por nome do cliente e matriz ou filial.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF2.1 Nível de acesso	A funcionalidade só pode ser executada por usuários com nível de administrador ou superior	Segurança	( )	(X)
NF2.2 Incluir Equipamento para Cliente	O sistema deve incluir o equipamento disponível para o cliente, que deve ser cadastrado por empresa, código, opção de locação ou venda, modelo e data.	Segurança	( )	(X)
NF2.3 Janela Única	Todas as funções relacionadas aos orçamentos devem ser efetuadas em uma única janela.	Interface	(X)	( )

Quadro 3 – Requisitos de cadastro equipamentos

O Quadro 4 contém o requisito funcional Ordem de serviço. Esse requisito funcional possibilitará ao administrador controle das ordens de serviço geradas pela empresa. Adicional ao requisito funcional está à lista de requisitos não-funcionais associados.

F3 Gerar Ordem de Serviço	**Oculto ( )**			
Descrição: A inclusão de uma ordem de serviço deve ser feita por um numero, pela empresa (filial, matriz), cliente e o técnico que atenderá a referida ordem. Além dos itens: data de entrada, data saída, valor dos serviços e peças utilizadas.				
Requisitos Não-Funcionais				
Nome	**Restrição**	**Categoria**	**Desejável**	**Permanente**
NF3.1 Nível de acesso	A funcionalidade só pode ser executada por usuários com nível de administrador ou técnico	Segurança	( )	(X)
NF3.2	O sistema deve fazer a inclusão	Segurança	( )	(X)

Incluir Ordem de Serviço	de uma ordem de serviço através de um código, nome da empresa(matriz, filial), cliente, técnico, data entrada e saída e valor total.			
NF3.3 Janela Única	Todas as funções relacionadas aos orçamentos devem ser efetuadas em uma única janela.	Interface	(X)	( )

Quadro 4 – Requisitos da ordem de serviço

O diagrama de classes descreve a visão estática do sistema em termos de classes e relacionamento entre elas. A Figura 9 representa o diagrama de classes do sistema com seus respectivos relacionamentos.

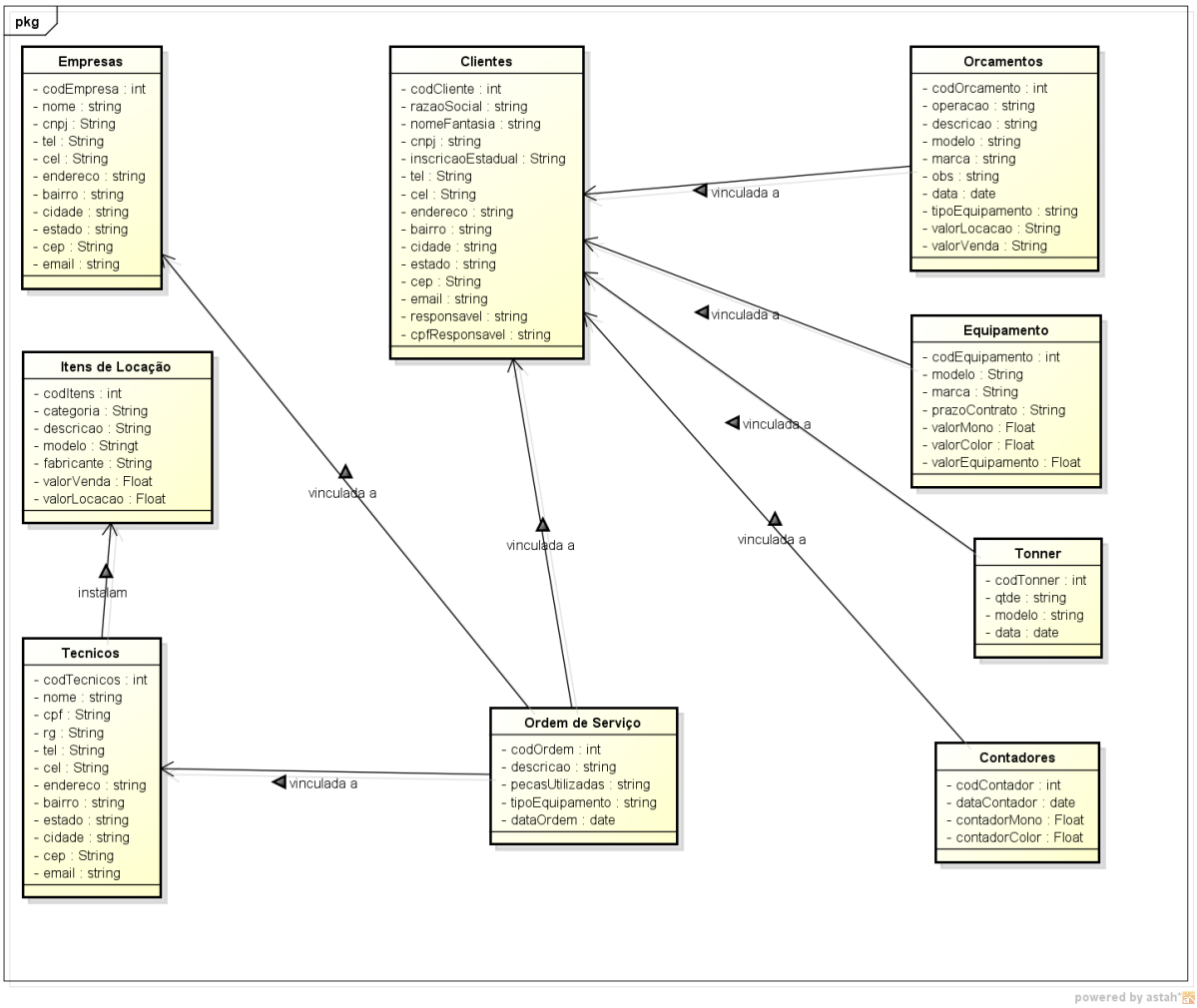


Figura 9 - Diagrama de classe

As tabelas definidas para o banco de dados estão listadas a seguir, nos Quadros 5 a 13. Elas representam as classes persistentes do sistema.

CLIENTES

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
CodCliente	Int	Não	Sim	Não
Nome	texto	Não	Não	Não
nomeFantasia	texto	Não	Não	Não
CNPJ	texto	Não	Não	Não
IE	texto	Não	Não	Não
Telefone	texto	Não	Não	Não
Celular	Texto	Não	Não	Não
Endereço	Texto	Não	Não	Não
Bairro	Texto	Não	Não	Não
Cidade	texto	Não	Não	Não
Estado	Texto	Não	Não	Não
Cep	Texto	Não	Não	Não
Email	Texto	Não	Não	Não
Responsável	texto	Não	Não	Não
cpfResponsavel	Texto	Não	Não	Não

Quadro 5 – Tabela de clientes

Empresas

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
CodEmpresa	Int	Não	Sim	Não
razaoSocial	texto	Não	Não	Não
nomeFantasia	texto	Não	Não	Não
CNPJ	texto	Não	Não	Não
IE	texto	Não	Não	Não
Telefone	texto	Não	Não	Não
Celular	Texto	Não	Não	Não
Endereço	Texto	Não	Não	Não
Bairro	Texto	Não	Não	Não
Cidade	texto	Não	Não	Não
Estado	Texto	Não	Não	Não

Cep	Texto	Não	Não	Não
Email	Texto	Não	Não	Não

Quadro 6 – Tabela de empresas

Equipamentos

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
CodEquipamento	Int	Não	Sim	Não
codCliente	Int	Não	Não	Sim
Modelo	texto	Não	Não	Não
Marca	texto	Não	Não	Não
prazoContrato	texto	Não	Não	Não
valorMono	Float	Não	Não	Não
valorColor	Float	Não	Não	Não
valorEquipamento	Float	Não	Não	Não

Quadro 7 – Tabela de equipamentos

Itens

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
CodItens	Int	Não	Sim	Não
Categoria	texto	Não	Não	Não
Descrição	texto	Não	Não	Não
Modelo	texto	Não	Não	Não
Fabricante	texto	Não	Não	Não
valorVenda	Float	Não	Não	Não
valorLocacao	Float	Não	Não	Não

Quadro 8 – Tabela de itens

Orçamentos

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
CodOrcamento	Int	Não	Sim	Não
codCliente	Int	Não	Não	Sim
Operação	Texto	Não	Não	Não
Descrição	Texto	Não	Não	Não
Marca	Texto	Não	Não	Não

Modelo	texto	Não	Não	Não
Obs	Texto	Não	Não	Não
Data	Date	Não	Não	Não
tipoEquipamento	Texto	Não	Não	Não
ValorVenda	Float	Não	Não	Não
ValorLocacao	Float	Não	Não	Não

Quadro 9 – Tabela de orçamentos

Técnicos

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
CodTecnico	Int	Não	Sim	Não
Nome	Texto	Não	Não	Não
Cpf	texto	Não	Não	Não
Rg	texto	Não	Não	Não
Telefone	texto	Não	Não	Não
Celular	texto	Não	Não	Não
Endereço	Texto	Não	Não	Não
Bairro	texto	Não	Não	Não
Cidade	Texto	Não	Não	Não
Estado	texto	Não	Não	Não
Cep	Texto	Não	Não	Não
Email	Texto	Não	Não	Não

Quadro 10 – Tabela de técnicos

Ordens Serviço

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
CodOrdem	Int	Não	Sim	Não
codCliente	Int	Não	Não	Sim
codTecnico	Int	Não	Não	Sim
codEmpresa	Int	Não	Não	Sim
Descrição	texto	Não	Não	Não
pecasUtilizadas	texto	Não	Não	Não
tipoEquipamento	Texto	Não	Não	Não

dataOrdem	Date	Não	Não	Não

Quadro 11 – Tabela de ordens de serviço

Toner

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
CodTonner	Int	Não	Sim	Não
codCliente	Int	Não	Não	Sim
Qtde	Texto	Não	Não	Não
Modelo	Texto	Não	Não	Não
Data	Date	Não	Não	Não

Quadro 12 – Tabela de toner

Contadores

Campo	Tipo	Nulo	Chave primária	Chave estrangeira
CodContador	Int	Não	Sim	Não
codCliente	Int	Não	Não	Sim
codEquipamento	Int	Não	Não	Sim
dataContador	Date	Não	Não	Não
contadorMono	Float	Não	Não	Não
contadorColor	Float	Não	Não	Não

Quadro 13 – Tabela de contadores

4.3 APRESENTAÇÃO DO SISTEMA

A Figura 10 apresenta os itens do menu cadastro que é composto por Clientes, Empresas, Equipamentos, Itens de Locação, Orçamentos, Técnicos, Ordem de Serviços, Toner, Contadores e a opção Sair.

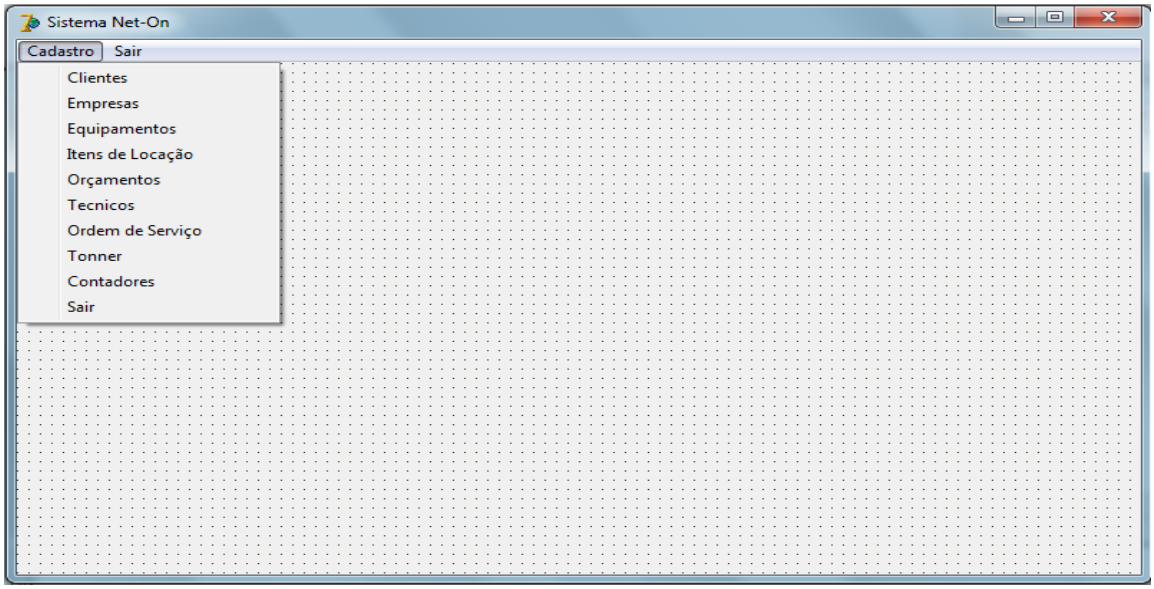


Figura 10 - Menu de cadastros

A Figura 11 mostra a tela de cadastro de clientes. Dessa tela podem ser acessados duas opções de relatório. Essa tela de cadastro possui as opções de código do cliente, nome, nome fantasia, cnpj, inscrição estadual, telefone, celular, endereço, bairro, cidade, estado, cep, e-mail, responsável pela empresa e cpf do mesmo. Na consulta de dados é possível visualizar todos os clientes cadastrados no banco de dados. Um registro (cliente) listado pode ser selecionado por meio de um clique ou por meio do botão localizar.

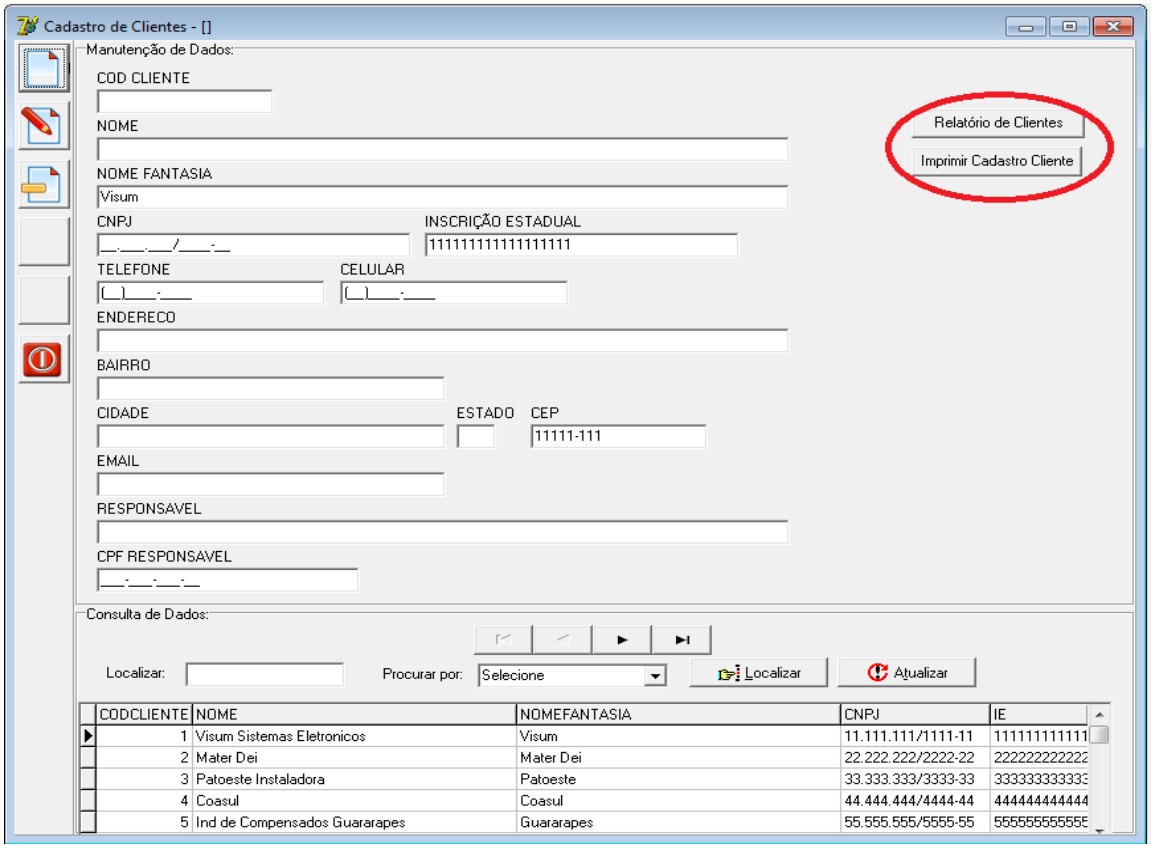
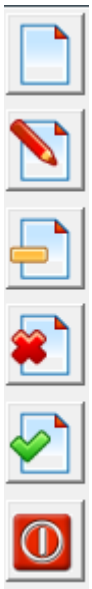


Figura 11 - Cadastro de clientes

A Figura 12 contém os botões que estão em todas as telas de cadastro do sistema. Esses botões se referem as operações de novo registro, alterar, excluir, cancelar, salvar e fechar.



Novo Documento

Alterar

Excluir

Cancelar

Salvar

Fechar

Figura 12 - Menu de opções – opções de armazenamento

Quando a respectiva tela de cadastro é acessada, apenas os botões Novo, Alterar, Excluir, Fechar e a consulta de dados estão habilitados. Após o usuário selecionar o botão Novo, as opções que estarão desabilitadas são: Novo, Cancelar e Excluir. Para excluir um lançamento basta selecionar no *groupbox* de consulta de dados um dos lançamentos desejados, automaticamente este é carregado para manutenção de dados (Figura 13).

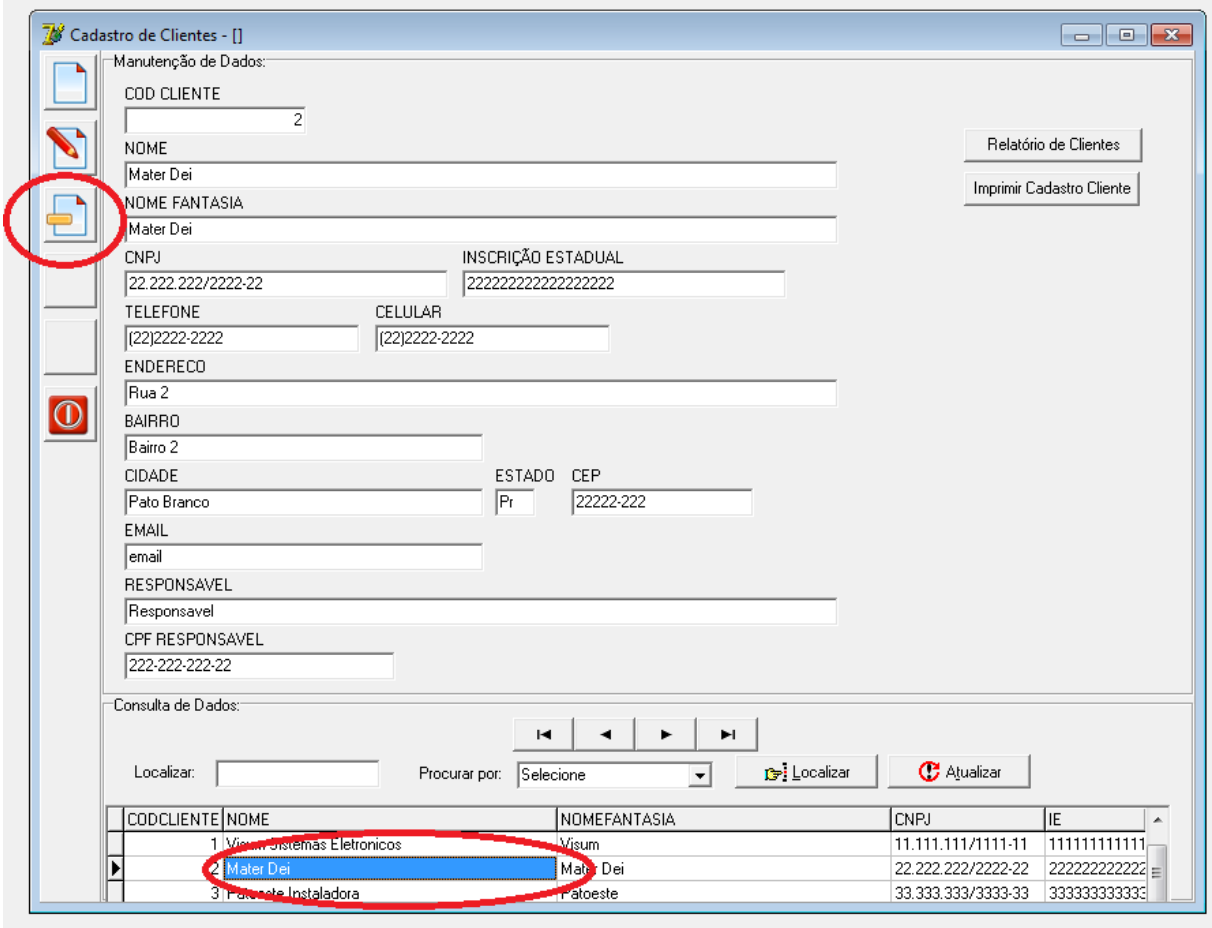
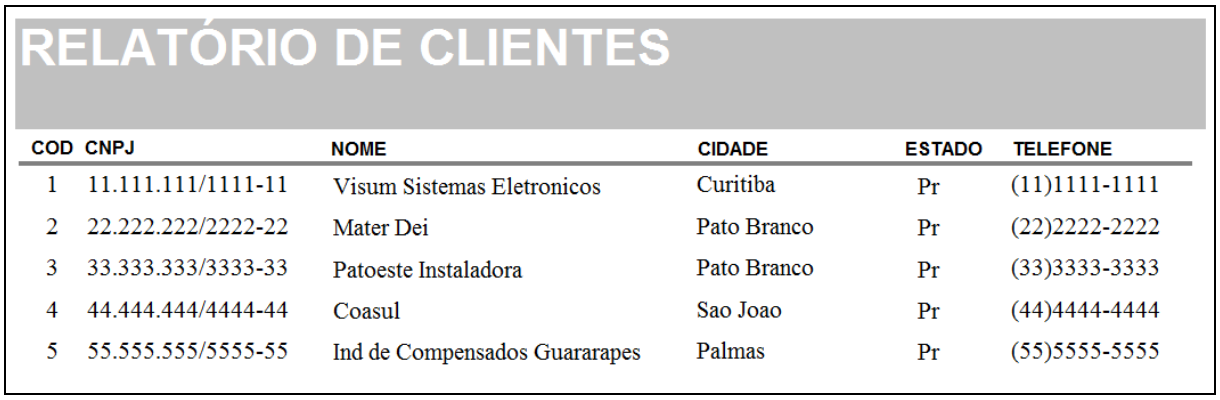


Figura 13 - Selecionar e excluir lançamentos

A Figura 14 apresenta o relatório de clientes relacionado na tela de cadastro de clientes. Esse relatório contém uma lista detalhada de todos os clientes cadastrados na empresa.

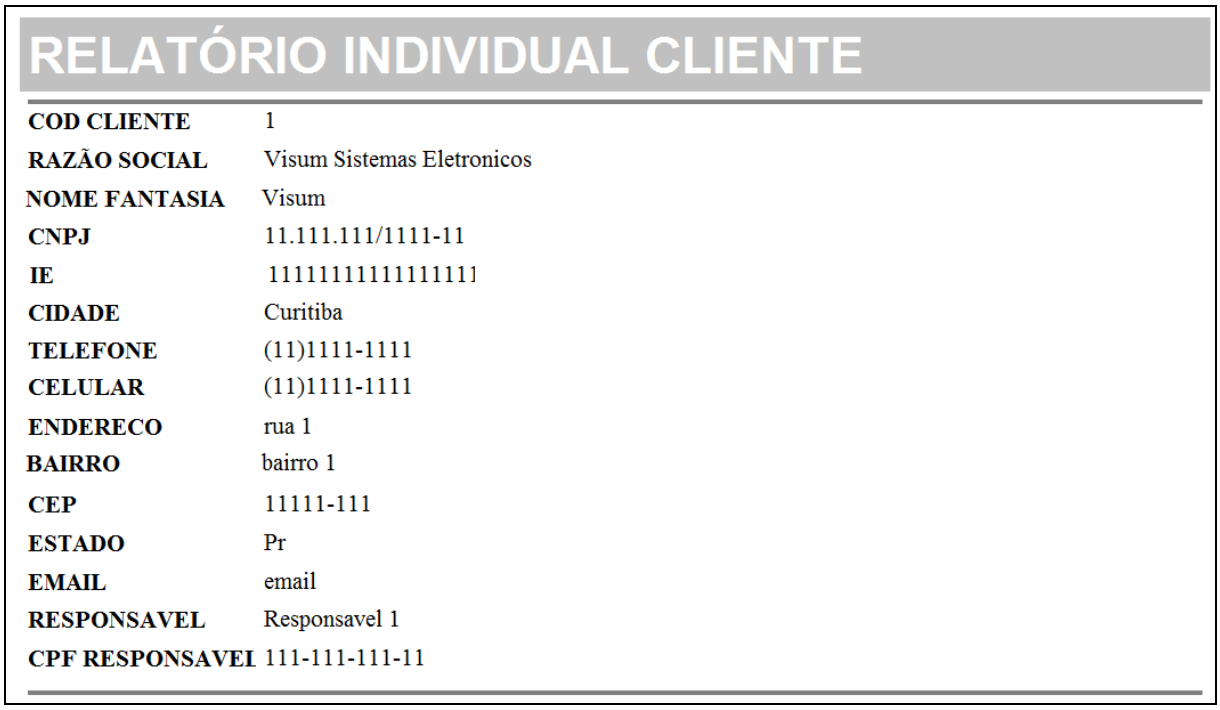


RELATÓRIO DE CLIENTES

COD	CNPJ	NOME	CIDADE	ESTADO	TELEFONE
1	11.111.111/1111-11	Visum Sistemas Eletronicos	Curitiba	Pr	(11)1111-1111
2	22.222.222/2222-22	Mater Dei	Pato Branco	Pr	(22)2222-2222
3	33.333.333/3333-33	Patoeste Instaladora	Pato Branco	Pr	(33)3333-3333
4	44.444.444/4444-44	Coasul	Sao Joao	Pr	(44)4444-4444
5	55.555.555/5555-55	Ind de Compensados Guararapes	Palmas	Pr	(55)5555-5555

Figura 3 - Relatório de clientes – lista

O relatório da Figura 15 é um exemplo de relatório individual de cada cliente cadastrado no sistema.



RELATÓRIO INDIVIDUAL CLIENTE

COD CLIENTE	1
RAZÃO SOCIAL	Visum Sistemas Eletronicos
NOME FANTASIA	Visum
CNPJ	11.111.111/1111-11
IE	1111111111111111
CIDADE	Curitiba
TELEFONE	(11)1111-1111
CELULAR	(11)1111-1111
ENDERECO	rua 1
BAIRRO	bairro 1
CEP	11111-111
ESTADO	Pr
EMAIL	email
RESPONSAVEL	Responsavel 1
CPF RESPONSAVEL	111-111-111-11

Figura 4 - Relatório individual de clientes

A Figura 16 apresenta outro exemplo de cadastro, de Ordem de serviço. Esse cadastro possui três caixas de combinação (*combobox*), nas quais serão carregados os dados cadastrados da tabela clientes, técnico e empresas, respectivamente.

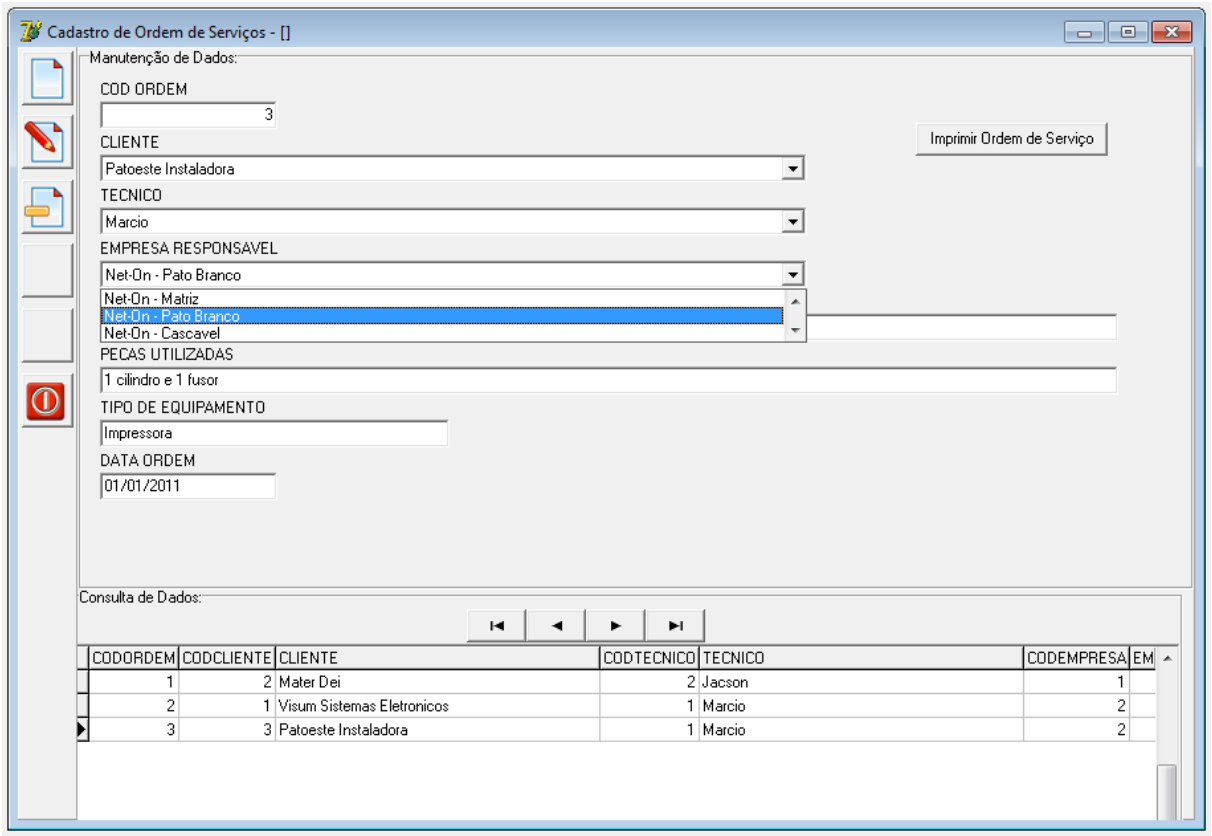


Figura 5 - Tela de cadastro de ordem de serviço

4.4 IMPLEMENTAÇÃO DO SISTEMA

Para exemplificar o código foram escolhidas as opções de cadastro de Cliente. O código do botão Novo é apresentado na Listagem 1. O código está no evento OnClick desse botão. Se o registro estiver em edição ou alteração, é feito o cancelamento e em seguida ela prepara para receber um novo registro, em que o Código do cliente recebe o foco.

```
procedure TfrmCadClientes.btnNovoClick(Sender: TObject);
begin
  inherited;
  EstadoBotoes('I');//Prepara botões para inclusão
  if ibtCliente.State in [dsEdit, dsInsert] then
    ibtCliente.Cancel; //Se a tabela estiver em edição ou alteração, cancela
  ibtCliente.Append; //Prepara a tabela para receber um registro novo
  edtCodCliente.SetFocus;//Seta o foco no componente de código
end;
```

Listagem 1 - Implementação botão novo – cadastro de clientes

Na Listagem 2 está o código do botão Alterar, no evento OnClick, o mesmo

prepara a tabela para edição por meio do “ibtCliente.Edit”, o Código do Cliente recebe falso pois não pode ser alterado e assim o campo Nome recebe o foco.

```
procedure TfrmCadClientes.btnAlterarClick(Sender: TObject);
begin
  inherited;
  EstadoBotoes('A');
  ibtCliente.Edit; //Prepara a tabela para alteração
  edtCodCliente.Enabled := false;
  edtNome.SetFocus; //Como o código estará desabilitado, seta o foco no nome
end;
```

Listagem 2 - Implementação botão alterar – cadastro de clientes

O botão excluir está configurado pelo evento OnClick. Quando clicado no botão excluir será mostrado ao usuário a mensagem: “Deseja excluir o registro selecionado?”, se sim o registro é excluído da tabela (Listagem 3).

```
procedure TfrmCadClientes.btnExcluirClick(Sender: TObject);
begin
  inherited;
  if Application.MessageBox('Deseja excluir o registro selecionado?',
    'Confirmação', MB_YESNO+MB_ICONQUESTION) = idYes then
    begin
    ibtCliente.Delete;
    EstadoBotoes('C');
  end;
end;
```

Listagem 3 - Implementação botão excluir – cadastro de clientes

Por meio do botão Cancelar, é possível cancelar um cadastro que está em edição. Isso é feito pelo comando “ibtCliente.Cancel” como mostra a Listagem 4.

```
procedure TfrmCadClientes.btnCancelarClick(Sender: TObject);
begin
  inherited;
  edtCodCliente.Enabled := True;
  ibtcliente.Cancel;
  EstadoBotoes('C');
end;
```

Listagem 4 - Implementação botão cancelar – cadastro de clientes

Para salvar, primeiramente é feito um laço passando em todos os campos da tabela, se algum registro não estiver preenchido, então é mostrada a mensagem ao usuário informando qual é esse campo e é colocando o foco no local que está com erro ou em branco. Na sequência os dados são gravados no banco pelo comando “ibtClientes.Post” e, assim, o Campo “Código” recebe o foco para um novo Cadastro

(Listagem 5).

```
procedure TfrmCadClientes.btnSalvarClick(Sender: TObject);
begin
  inherited;
    try
    if edtCodCliente.Text = '' then
      begin
      ShowMessage('Informe o código!');
      edtCodCliente.SetFocus;
      end
    else if edtNome.Text = '' then
      begin
      ShowMessage('Informe o nome!');
      edtNome.SetFocus;
      end
    else if edtCNPJ.Text = '' then
      begin
      ShowMessage('Informe o CNPJ!');
      edtCNPJ.SetFocus;
      end
    else if edtTel.Text = '' then
      begin
      ShowMessage('Informe o Telefone!');
      edtTel.SetFocus;
      end
    else if edtCel.Text = '' then
      begin
      ShowMessage('Informe o Celular!');
      edtCel.SetFocus;
      end
    else if edtEndereco.Text = '' then
      begin
      ShowMessage('Informe o Endereco!');
      edtEndereco.SetFocus;
      end
```

```
    else if edtBairro.Text = '' then
      begin
      ShowMessage('Informe o Bairro!');
      edtBairro.SetFocus;
      end
    else if edtEstado.Text = '' then
      begin
      ShowMessage('Informe o Estado!');
      edtEstado.SetFocus;
      end
    else if edtCidade.Text = '' then
      begin
      ShowMessage('Informe a Cidade!');
      edtCidade.SetFocus;
      end
    else if edtCEP.Text = '' then
      begin
      ShowMessage('Informe o CEP!');
      edtCEP.SetFocus;
      end
    else if edtEmail.Text = '' then
      begin
      ShowMessage('Informe o Email!');
      edtEmail.SetFocus;
      end
    else if edtResponsavel.Text = '' then
      begin
      ShowMessage('Informe o Responsavel pela Empresa!');
      edtResponsavel.SetFocus;
      end
    else if edtCPFResponsavel.Text = '' then
      begin
      ShowMessage('Informe o CPF do Responsavel pela Empresa!');
      edtCPFResponsavel.SetFocus;
      end
    else
      begin
      ibtCliente.Post; //Grava inclusão ou alteração
      EstadoBotoes('C'); //Prepara botões para uma nova operação
      edtCodCliente.Enabled := True; //Habilita o campo código
      end;
  except
    on E: Exception do
      ShowMessage('Erro ao gravar o registro: '+E.Message);
  end;
end;
```

Listagem 5 - Implementação botão salvar – cadastro de clientes

A Listagem 6, mostra uma *procedure* padrão em todo o sistema para limpar os campos depois de salvo no banco de dados, nela cada “edit” recebe o comando “clear”.

```
procedure TfrmCadClientes.LimparCampos;
begin
  edtCodCliente.Clear;
  edtNome.Clear;
  edtCNPJ.Clear;
  edtTel.Clear;
  edtCel.Clear;
  edtEndereco.Clear;
  edtBairro.Clear;
  edtEstado.Clear;
  edtCidade.Clear;
  edtEmail.Clear;
  edtResponsavel.Clear;
  edtCPFResponsavel.Clear;
end;
```

Listagem 6 - Implementação limpar campos – cadastro de clientes

Na Listagem 7 é mostrado o Evento “OnShow” do formulário de Cadastro de Clientes. Nesse, primeiramente é executado o comando de Limpar os Campos do formulário e em seguida abre a Tela de Cadastro de Clientes para receber os dados.

```
procedure TfrmCadClientes.FormShow(Sender: TObject);
begin
  inherited;
  cCaption := 'Cadastro de Clientes';
  EstadoBotoes('C'); //Prepara botões para receber alguma operação
  LimparCampos;
  ibtCliente.Open; //Abre a tabela
end;
```

Listagem 7 - Implementação evento formshow – cadastro de clientes

Da mesma maneira na Listagem 8 é mostrado o Evento “OnClose”, que fecha a tabela após o seu uso.

```
procedure TfrmCadClientes.FormClose(Sender: TObject;
  var Action: TCloseAction);
begin
  ibtCliente.Close; //Fecha a tabela
  inherited;
  frmCadClientes := nil;
end;
```

Listagem 8 - Implementação evento formclose – cadastro de clientes

5 CONCLUSÃO

Este trabalho teve como objetivo apresentar a modelagem e o desenvolvimento de um sistema para locação de equipamentos de Informática para uma empresa em específico, utilizando a linguagem Delphi e o banco de dados Firebird. Contudo, o sistema pode ser utilizado por empresas que tenham características semelhantes à empresa para a qual o sistema foi desenvolvido.

O desenvolvimento deste trabalho permitiu verificar que o uso de uma ferramenta de análise orientada a objetos e UML traz facilidades na modelagem de um sistema. Isso porque os modelos, diagramas, produzidos permitem organizar a solução do problema de formas distintas. E a utilização de modelos para a elaboração de um *software* é de fundamental para entender o problema e a solução computacional proposta.

A UML constitui uma linguagem padrão de modelagem de sistemas e que nos permite elaborar, de forma teórica e gráfica, toda a documentação e construção de artefatos gráficos de um *software*. E a ferramenta *Astah* possibilita apresentar a modelagem orientada a objetos, utilizando os diagramas da UML.

A ideia de desenvolver um sistema para gerenciamento de impressão, decorreu das dificuldades encontradas no dia-a-dia da empresa utilizada como estudo de caso com planilhas e documentos feito de maneira inadequada. Gerenciar vários tipos de equipamentos e suprimentos não é fácil. E portar isso para um sistema gerou diversas dificuldades, tanto na modelagem quanto na implementação.

Apesar das dificuldades encontradas que foram superadas com o desenvolvimento do sistema em partes, iniciando pelos cadastros mais simples, o sistema foi desenvolvido como planejado. Está em uso na empresa e possivelmente necessidades de ajustes sejam identificadas. Esses ajustes serão implementados.

REFERÊNCIAS

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

CANTU, M., **Dominando o Delphi 6 "A Bíblia".** São Paulo. Makron Books, 2000.

CANTU, 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 de outubro de 2010.

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

IBEXPERT. **IBExpert dveloper sudio**. Disponível em: <http://www.ibexpert.com/>. Acesso em 14 de março de 2011.

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

POTOK T.; MLADEN V. **Development productivity for commercial software using object-oriented methods**. In: The 1995 conference of the Centre for Advanced Studies on Collaborative research, Toronto, Ontario, Canada, p. 52-59, 1995.

REPORT BUILDER. **Getting started with report builder 3.0**. Disponível em: <http://technet.microsoft.com/en-us/library/dd220460.aspx>. Acesso 20 de maio de 2011.

RUMBAUGH, J. et al. **Modelagem e projeto baseado em objeto**. Rio de Janeiro: Campus, 1997.

SILVA FILHO, A. M. **Introdução à programação orientada a objetos.** Disponível em: <http://www.espacoacademico.com.br/035/35amsf.htm>. Acesso em 10/03/2011.

WAZLAWICK, R. S. **Análise e projeto de sistemas de informação orientados a objetos.** Rio de Janeiro: Elsevier Editora Ltda., 2004.