FACULDADES INTEGRADAS DE TAQUARA

CURSO DE SISTEMAS DE INFORMAÇÃO

SISTEMA DE FLUXO DE CAIXA UTILIZANDO ARQUITETURA ORIENTADA A SERVIÇO

LUCIANO HOFFMAISTER RIBEIRO

Taquara

2009

LUCIANO HOFFMAISTER RIBEIRO

SISTEMA DE FLUXO DE CAIXA UTILIZANDO ARQUITETURA ORIENTADA A SERVIÇO

Trabalho de conclusão de curso apresentado ao Curso de Sistemas de Informação das Faculdades Integradas de Taquara, sob orientação do Professor M. Eng. Marcelo de Cunha.

Taquara

2009

AGRADECIMENTOS

Agradeço a Deus por estar ao meu lado dando-me saúde para continuar atingindo meus objetivos. Aos meus pais, João e Cirlei, o incentivo para buscar o conhecimento, apoiando as minhas decisões. Aos meus familiares e amigos, o suporte necessário nas ocasiões mais difíceis e compreensão em momentos que deixei de compartilhar ao lado deles. Aos meus colegas, com quem sempre pude contar, possibilitando a troca de conhecimentos para alcançar mais esta etapa. Aos professores da instituição, o aprendizado adquirido, em especial ao Marcelo Azambuja, que contribuiu imensamente para a conclusão deste trabalho.

Agradeço ao professor Sérgio Nikolay e ao amigo Roberto Villas Bôas por terem compartilhado os seus conhecimentos administrativos e financeiros.

Um especial agradecimento à minha esposa Astrid e à minha filha Laura, que tiveram a paciência necessária para continuar ao meu lado compreendendo que os momentos que não pude conviver ao lado delas foi passageiro. Amo muito vocês.

RESUMO

A administração financeira requer ferramentas que facilitem a análise dos dados da empresa e que ajudem os gestores e gerentes a avaliarem a situação financeira, assim como possibilitar a análise das previsões de desembolso e recebimentos. Neste sentido, o fluxo de caixa é uma ferramenta essencial às empresas, pois facilita a análise da situação financeira e pode ser considerado um dos principais instrumentos de análise e avaliação de uma organização. Os *softwares* que contemplam estas necessidades estão disponíveis em sistemas integrados de gestão empresarial de grande porte e representam um custo elevado de implantação para empresas de médio e pequeno porte. Este trabalho tem como objetivo oferecer um sistema de fluxo de caixa que facilite a análise gerencial e financeira de empresas, permitindo que os gestores tenham uma ferramenta de fácil manuseio, disponibilizando a análise dos valores realizados e previstos com comparativos em determinados períodos de forma sintética e analítica. Para isto a utilização da arquitetura orientada a serviços permite a execução remota do aplicativo e oferece uma rápida integração com sistemas de gestão empresarial.

Palavras-chave: Fluxo de Caixa. Arquitetura Orientada a Serviços. *Web Service*. Sistema de Apoio. Gestão Financeira.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

A administração financeira requer ferramentas que facilitem a análise dos dados da empresa e que ajudem os gestores e gerentes a avaliarem a situação financeira, assim como possibilitar análise das previsões de desembolso e recebimentos, além da análise do passado, através daquilo que já foi realizado.

Outra funcionalidade que facilita a gestão do negócio é a possibilidade de análises variadas e detalhadas em relação às projeções futuras e ponderações do que realmente se efetivou em um determinado dia, semana, mês ou ano. Outro ponto importante a ser destacado é que o sistema de apoio ao gestor financeiro deve contemplar as análises das informações em forma de gráficos comparativos de períodos atuais com os mesmos períodos de anos anteriores, traçando assim uma análise de ponderação média, sempre com as informações atualizadas, vindas da base de dados sem defasagem de tempo a fim de não distorcer qualquer tipo de análise.

Já existem *softwares* (programas de computador) que contemplam estas necessidades, mas normalmente estão disponíveis em sistemas integrados de gestão empresarial de grande porte, ERP (*Enterprise Resource Planning* – sistema de gestão empresarial), como SAP (*Systems, Applications and Products* – empresa alemã de soluções de ERP), Oracle ou TOTVS (empresa brasileira de soluções de ERP), mas que podem representar um custo elevado de implantação para empresas de médio e pequeno porte. Normalmente os maiores custos estão ligados a treinamento, importação, análise de dados e consultoria, além da aquisição do *software*.

Uma alternativa encontrada pelas pequenas empresas é o controle manual de suas informações financeiras, por intermédio de fluxo de caixa em planilhas eletrônicas, o que permite uma flexibilidade quanto à entrada e manuseio dos dados, mas em contrapartida dificulta a integração entre a equipe de trabalho e as demais áreas da empresa, pois a interligação com os demais sistemas informatizados fica dificultada em função da incompatibilidade entre os mesmos, necessitando intercâmbio dos dados e não disponibilizando o acesso *online* às informações.

1.1 Justificativa

De acordo com o descrito anteriormente, o fluxo de caixa é uma ferramenta essencial às empresas, pois facilita a análise da situação financeira. Ele pode ser considerado um dos principais instrumentos de análise e avaliação de uma organização, proporcionando ao administrador uma visão futura dos recursos financeiros da empresa seja ela de qualquer porte. O fluxo de caixa em muitos casos não é bem explorado pelos sistemas disponíveis no mercado, que muitas vezes contemplam somente as entradas e saídas de recursos, sem oferecer consultas analíticas ou comparativos entre as rubricas envolvidas.

Neste contexto a ferramenta desenvolvida não é um sistema de retaguarda, que contemple as rotinas básicas de um módulo financeiro, tais como contas a pagar e receber, controle de caixa, cobrança, etc. Essas funcionalidades são comuns em quase todos os sistemas de gestão, que priorizam a execução dessas tarefas, que são essenciais, mas voltadas para sanar tarefas burocráticas, como controlar inadimplência de clientes, remessa de cobrança a bancos, quitar obrigação com fornecedores, entre outras, e com pouco enfoque no apoio aos administradores. Normalmente os sistemas de retaguarda preocupam-se em apresentar de forma resumida o que se tem a receber e a pagar em um determinado período.

As funcionalidades já existentes no sistema de retaguarda, principalmente as entradas e desembolsos financeiros, irão formar a base da ferramenta de fluxo de caixa. Assim, o aplicativo estará focado na análise do negócio, permitindo que o administrador que irá tomar as decisões possa realizar diferentes pesquisas, analisando os dados de forma sintética e analítica. A seleção de diferentes períodos, com manutenção nos valores apresentados de forma simples e rápida, permitirá a visualização da informação de diferentes formas, realizando comparativos entre diferentes rubricas, períodos e demais informações de maior relevância para a análise do negócio.

O diferencial que visa o presente aplicativo é a possibilidade de integrar a ferramenta com diferentes sistemas de gestão, por intermédio da tecnologia SOA (*Service-oriented Architecture* – arquitetura orientada a serviço), usando *Web Services* (serviços Web) para comunicar com a base de dados da empresa e, a partir deste, prover as informações necessárias à aplicação cliente de fluxo de caixa.

Nesta situação, para diferentes sistemas de gestão, há a necessidade de configurar somente o servidor, serviço de acesso aos dados (*web service*), que estará disponível na *intranet* da empresa, permanecendo o aplicativo cliente sem modificação para diferentes situações. Na Figura 1, tem-se uma visão geral do sistema, onde se verifica que o aplicativo cliente não acessa diretamente os dados do sistema de retaguarda da empresa, tampouco o repositório com os dados dos lançamentos complementares ao fluxo de caixa. Os dados são fornecidos pelos *Web Services* hospedados no servidor e estes são disponibilizados para a aplicação cliente do fluxo de caixa que faz a manipulação das informações e então as apresenta ao usuário da forma que for mais adequada.

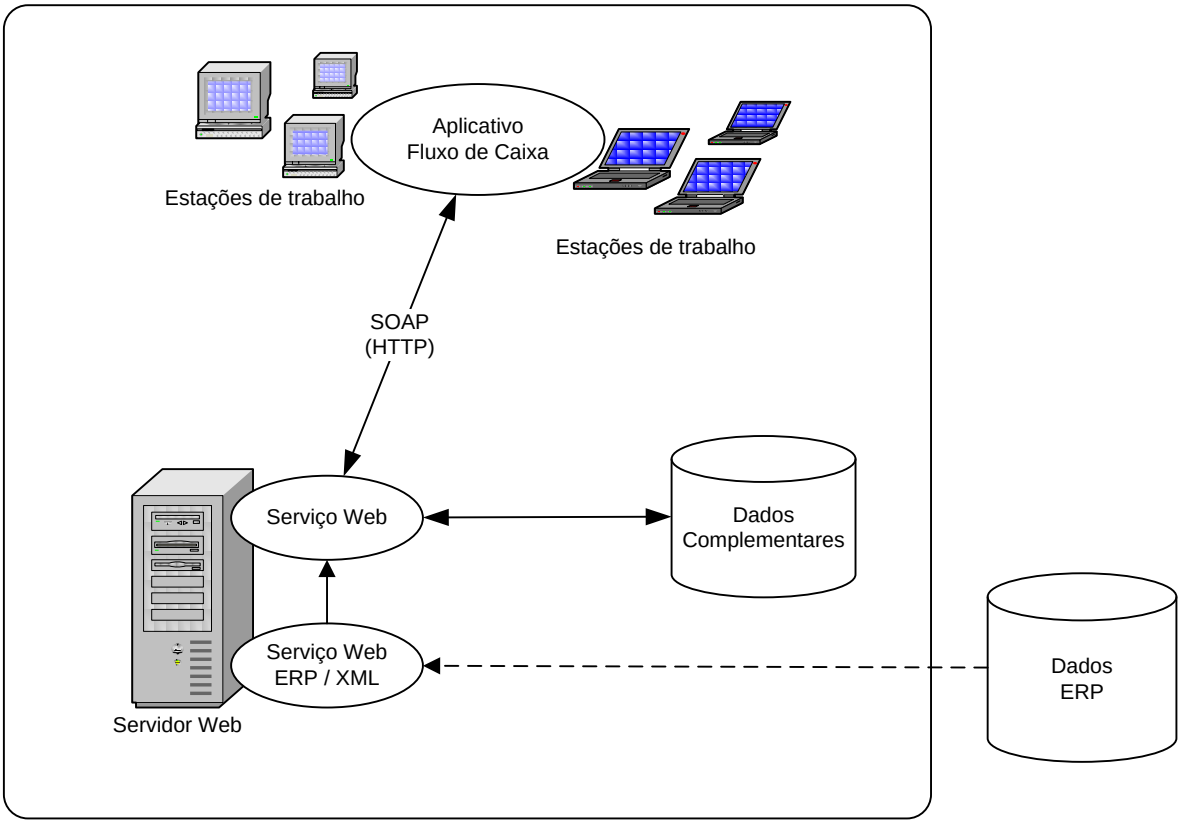


Figura 1 – Visão geral do sistema

1.2 Objetivos

1.2.1 Objetivos gerais

Desenvolver e implementar um *software* de fluxo de caixa que facilite a análise gerencial e financeira de empresas, permitindo que os gestores tenham uma ferramenta de fácil manuseio e que possibilite a manutenção dos valores e rubricas envolvidas de forma rápida e simples, disponibilizando a análise dos valores realizados e previstos com comparativos em determinados períodos.

1.2.2 Objetivos específicos

a) desenvolver um servidor de aplicativo que se encarregue da comunicação com o sistema de retaguarda;
b) desenvolver um aplicativo cliente, que será a *interface* com o usuário, apresentando as informações requisitadas;
c) possibilitar que o usuário possa realizar ajustes e modificações nas rubricas e valores envolvidos;
d) permitir que sejam emitidos relatórios com valores realizados e previstos, podendo selecionar períodos distintos;
e) disponibilizar funcionalidade para realizar comparativos entre rubricas de diferentes períodos;
f) visualizar os comparativos na forma de gráficos;
g) possibilitar a exibição do fluxo de caixa sintético e analítico.

1.3 Organização do trabalho

O presente trabalho está organizado em seis seções, sendo elas:

a) a seção 1 apresentou uma introdução, justificativa e objetivo trabalho;
b) na seção 2 é realizada a fundamentação teórica dos principais conceitos envolvidos;
c) a seção 3 contempla a metodologia utilizada no desenvolvimento;
d) na seção 4 são explicadas as tecnologias empregadas;
e) a seção 5 apresenta os resultados obtidos com o sistema desenvolvido;
f) a seção 6 expõe as conclusões do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

O princípio do aplicativo é que ele possa trabalhar em conjunto com outros sistemas de retaguarda. Neste caso ele seria executado como um novo módulo que possa ser integrado, desde que seja fornecido o acesso necessário à base de dados para a sua implementação. Sendo assim, são necessárias algumas tecnologias e ferramentas que necessitam ser contextualizadas. São elas:

a) Fluxo de Caixa;
b) SOA – *Service-Oriented Architecture*;
c) Servidor HTTP;
d) *Web Service*;
e) Banco de Dados.

2.1 Fluxo de Caixa

Segundo Ross (2000, p. 63) “[...] Por fluxo de caixa queremos simplesmente dizer a diferença entre a quantidade de dólares que entrou no caixa e a quantidade de dólares que saiu”. De uma maneira simplista, podemos dizer que o principal interesse dos proprietários da empresa está em saber quanto de dinheiro há disponível.

Um conceito de fluxo de caixa é apresentado por Zdanowicz (1998, p. 40):

> Denomina-se fluxo de caixa de uma empresa ao conjunto de ingressos e desembolsos de numerário ao longo de um período determinado. O fluxo de caixa consiste na representação dinâmica da situação financeira de uma empresa, considerando todas as fontes de recursos e todas as aplicações em itens do ativo.
> De forma mais sintética pode-se conceituar: é o instrumento de programação financeira, que corresponde às estimativas de entradas e saídas de caixa em certo período de tempo projetado.

Netto (1999, p.92), define fluxo de caixa como sendo:

> [...] o saldo aritmético entre entradas e saídas de moeda no caixa a cada instante, realizado e/ou projetado durante um determinado período, proveniente do movimento operacional da empresa.

Considerando-se que uma empresa realiza diversas operações diariamente,

dentre elas, compras, vendas e investimentos, com grande frequência o volume de entradas e saídas pode variar, assim como o mercado pode reagir de formas diferentes aos produtos e/ou serviços que são oferecidos pela empresa.

Sendo assim, para que a empresa possa estar em dia com suas obrigações, é essencial que no vencimento das mesmas ela tenha condições de quitar suas dívidas, ou seja, os administradores devem estar sempre interessados na disponibilidade de caixa, bancos e aplicações financeiras.

Segundo Zdanowicz (1998, p. 38):

> Assim, a projeção de fluxo de caixa depende de vários fatores como tipo de atividade econômica, o porte da empresa, o processo de produção e/ou comercialização, se é contínuo ou não, etc. Deve-se considerar, também, as fontes de caixa que podem ser internas e/ou externas. Os ingressos decorrentes de fontes internas podem ser originados por vendas à vista, cobranças das vendas a prazo, vendas de itens do ativo permanente, enquanto as fontes externas são identificadas como provenientes de fornecedores, instituições financeiras e governo.

O fluxo de caixa não é necessariamente um relatório de apresentação obrigatória, de acordo com as leis nacionais, mas deve ser encarado como um relatório gerencial de uso interno pela empresa, que na maioria dos casos é amplamente utilizado em pequenas e médias empresas que fazem uso do fluxo de caixa na administração (NETTO, 1999).

Outro ponto que é importante ser ressaltado, de acordo com Zdanowicz (1998), é que nem sempre as vendas ocorrem em períodos regulares, é comum, dependendo da atividade econômica, que a empresa receba seus recursos em determinados período do ano, se caracterizando vendas sazonais, por questão de moda, safra ou estação. Também é comum que suas receitas aconteçam em períodos maiores, por força de contratos firmados, como os casos de construção civil ou naval, por exemplo.

Netto (1999, p. 93) entende que o fluxo de caixa “[...] principalmente em grandes empresas, é o instrumento principal do administrador financeiro, pois é o relatório que vai orientá-lo nas necessidades de aplicação e obtenção de recursos para manter o coração da empresa batendo [...]”.

De qualquer forma, o que realmente importa para a empresa e seus administradores, é que após ocorrer a receita, a mesma se caracterize em real entrada no caixa, pois são necessárias para cobrir os desembolsos de caixa, que segundo Zdanowicz (1998, p. 38) “[...] podem ser classificados como regulares, periódicos e irregulares.”, sendo que os desembolsos regulares normalmente tem

maior peso e caracterizados por folha de pagamento, fornecedores, despesas administrativas e de vendas.

Neste contexto, o fluxo de caixa torna-se uma ferramenta essencial, pois um de seus principais objetivos é possibilitar aos administradores "[...] uma visão das atividades desenvolvidas, bem como as operações financeiras que são realizadas diariamente, no grupo do ativo circulante, dentro das disponibilidades, e que representam o grau de liquidez da empresa" (ZDANOWICZ, 1998, p. 41). Ou seja, projetar os recebimentos e desembolsos dos recursos financeiros em um determinado período para que seja possível analisar as necessidades para suprir o saldo de caixa, assim como, permitir a aplicação desses recursos em caso de excedentes no período analisado.

Outro ponto ressaltado por Netto:

> O erro básico e sistemático que temos encontrado nas empresas que assessoramos é insistir em administrar a mesma pelo fluxo de caixa, que na realidade utiliza o relatório para saber o que aconteceu, e não o que vai ou pode acontecer, utilizando-o como uma autópsia, e não como um diagnóstico de um provável problema [...] (1999, p. 94).

Isto reforça a necessidade de que para uma correta análise é importante que o fluxo de caixa apresente as informações passadas, ou seja, o que foi efetivamente realizado, mas não se pode abrir mão da possibilidade de procurar antever possíveis problemas futuros, apresentando uma previsão dos recebimentos e desembolsos de caixa de maneira sintética e analítica, facilitando a análise da gestão da empresa.

2.2 SOA – Service-Oriented Architecture

SOA (*Service-Oriented Architecture* - arquitetura orientada a serviços) pode ser entendida como um paradigma ou uma arquitetura de *software* onde as funcionalidades que são implementadas nos aplicativos, estão disponíveis sob forma de serviços, que podem ser estruturados em barramentos que irão prover as *interfaces* de acesso aos demais aplicativos.

Nascimento (2007, p. 16) conceitua SOA como sendo "[...] um conceito no qual aplicativos e/ou rotinas são disponibilizados como serviços em uma rede (*Intranet/Internet*) de forma independente, se comunicando através de padrões

abertos.”

Para Josuttis (2008, p. 7) SOA, “[...] é um paradigma para a realização e a manutenção dos processos corporativos que se encontram em grandes sistemas distribuídos.”

Josuttis (2008, p. 12) complementa:

> SOA não é uma arquitetura concreta: é algo que conduz a uma arquitetura concreta. Você pode chamá-la de estilo, paradigma, conceito, perspectiva, filosofia ou representação. Ou seja, SOA não é uma ferramenta ou *framework* que se possa comprar. É uma abordagem, uma maneira de pensar, um sistema de valores que leva a certas decisões concretas quando se projeta uma concreta arquitetura de *software*.

Deve-se enfatizar que a arquitetura orientada a serviço se trata de uma técnica de desenvolvimento de aplicativos, permitindo que o sistema seja dividido em camadas distintas, onde cada uma das camadas do sistema é responsável por uma tarefa do aplicativo.

Benedete Junior (2007, p. 7) entende que “[...] SOA, como o nome diz, é uma “arquitetura”, ou seja, um conjunto de princípios, padrões e orientações que englobam desde uma visão de negócio até as possíveis alternativas tecnológicas.”

Outro ponto que deve ser ressaltado é que SOA permite a fácil integração entre sistemas, principalmente sistemas legados, possibilitando o acoplamento de novos módulos utilizando esse paradigma, fazendo uso de serviços que irão realizar a comunicação com um único repositório de dados, isto por que grandes sistemas utilizam plataformas diferentes e com diferentes linguagens de programação. Nesta situação dificilmente se consegue uma harmonia entre estas diferentes tecnologias, como analisado por Josuttis (2008). Ou seja, sob esse aspecto, fica evidenciada a interoperabilidade entre os sistemas, sejam eles de qualquer linguagem ou plataforma, quanto ao reuso de aplicações, pois os aplicativos desenvolvidos com estas características são facilmente reutilizáveis em aplicativos futuros.

SOA tem uma abordagem que admite a heterogeneidade dos grandes sistemas, ou seja, que realmente estes sistemas fazem uso de diferentes plataformas, linguagens e paradigmas de programação, para possibilitar a sua integração, além de utilizar funcionalidades distribuídas que podem estar em diferentes locais, quanto ao seu domínio de propriedade.

Josuttis (2008, p. 14), entende que:

> Essa é uma das ideias-chave de SOA e ela pode dar à SOA o poder para iniciar uma revolução. Semelhante aos métodos “ágeis” de desenvolvimento de software, que aceitam que os requisitos mudem em vez de ficar lutando contra essas mudanças. SOA simplesmente aceita que existe heterogeneidade nos grandes sistemas. Isso leva a uma maneira de pensar bem diferente e, de repente, temos uma maneira de lidar com os grandes sistemas distribuídos que realmente funciona.

Nascimento (2007) entende que SOA utiliza uma abordagem que busca padronizar alguns conceitos de desenvolvimento de *software*, como modularidade, componentização e *interfaces* de aplicação, sendo comuns à orientação a objetos, mas de uma maneira diferente, onde seja possível utilizar funções genéricas únicas em cada aplicação, de forma que possam ser reutilizáveis e compartilhadas para acesso *online*.

Benedete Junior (2007, p. 28), ressalta outro ponto importante:

> [...] SOA é a busca por uma linguagem comum entre as áreas de negócio e TI. Normalmente, o negócio contrata serviços da área de TI, e não se preocupa com os detalhes de como os sistemas são construídos (e nem suas limitações). De forma similar, a TI constrói as aplicações se baseando apenas nas especificações recebidas, sem uma visão mais abrangente do impacto no negócio. Cada equipe fica em seu domínio de conhecimento, o que limita a troca de informações e o desenvolvimento colaborativo.

Na Figura 2 podemos verificar que a integração de diferentes aplicações em um ambiente SOA ocorre por intermédio da *interface* de serviço.

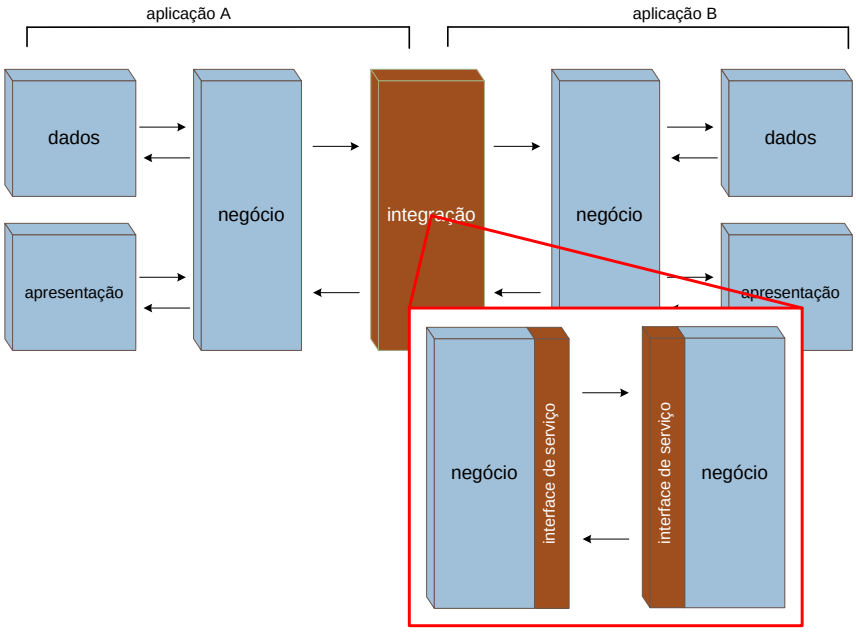


Figura 2 – Representação lógica de integração da arquitetura orientada a serviço
Adaptado de Erl (2004)

SOA está baseada em três características: serviços, interoperabilidade e acoplamento fraco, as quais serão identificadas a seguir.

2.2.1 Serviços

Em se tratando de uma arquitetura orientada a serviço, o termo serviço é de fundamental importância, pois é uma das bases de SOA. Existem diferentes definições para o termo “serviço”, pois pode abranger diferentes áreas.

Conforme descrito por Benedete Junior (2007, p. 20):

> Do ponto de vista do negócio, são as funcionalidades providas pela empresa para seus clientes e parceiros, por exemplo, um serviço de saque, um serviço de abertura de contas. Do ponto de vista de TI, trata-se de um componente de aplicação cujas funcionalidades estão disponíveis para outros sistemas ou usuários.

Josuttis (2008, p. 15), argumenta que “[...] pode-se considerar um serviço como sendo uma representação da TI de uma funcionalidade de negócio independente”, ou seja, qualquer funcionalidade do sistema pode ser tratada como um serviço, seja criar um cliente em sua base de dados, consultar os pedidos pendentes de entrega em um determinado período ou relacionar as previsões de receita num período de um determinado cliente.

Josuttis (2008) entende a importância de serviço como uma funcionalidade do negócio de tal maneira que, dependendo da sua aplicação, possa representar uma tarefa independente do sistema que seja correspondida a uma atividade no mundo real, onde há a real necessidade de as pessoas entenderem o que realmente um serviço é capaz de fazer e a sua aplicabilidade.

Do ponto de vista técnico, o serviço é uma *interface* que irá interagir com o sistema, por intermédio de troca de mensagens (envio e recebimento) que irão realizar uma determinada tarefa e/ou retornar alguma informação, Josuttis (2008).

2.2.2 Interoperabilidade

O conceito de interoperabilidade pode ser entendido como a capacidade de troca de informações entre sistemas diferentes de forma transparente, ou seja, sem que eles utilizem a mesma linguagem de programação ou a mesma plataforma e, independente disto, conseguirão estabelecer comunicação entre si.

A interoperabilidade é que permite que aplicativos diferentes possam trabalhar em conjunto, de maneira distribuída ou não. Sendo assim, uma funcionalidade de um sistema pode ser desenvolvida em linguagem Java e ser acessível por outros módulos do sistema ou até mesmo sistemas diferentes, sem que necessariamente sejam também desenvolvidas com a mesma linguagem.

2.2.3 Acoplamento fraco

Segundo Josuttis (2008, p. 16):

> [...] O acoplamento fraco é o conceito de minimizar as dependências. Quando as dependências estão minimizadas, as modificações têm os efeitos minimizados e os sistemas ainda executam quando partes deles estão quebradas ou indisponíveis.

De acordo com Josuttis (2008) o acoplamento fraco está ligado diretamente à capacidade do sistema quanto à flexibilidade, escalabilidade e tolerância a falhas. Neste sentido, o sistema necessita continuar em funcionamento quando uma de suas funcionalidades não estiver disponível.

Acoplamento fraco também diz respeito à capacidade de escalabilidade do sistema, ou seja, um sistema robusto deve “[...] evitar gargalos; caso contrário, crescer pode se tornar muito caro. Evitar gargalos é importante tanto do ponto de vista técnico quanto do organizacional” (JOSUTTIS, 2008, p. 16).

Em um sistema que contenha pouca dependência, ou seja, que possa ser flexível, tolerante à falhas e escalável, as modificações necessárias ou falhas que possam acontecer irão apresentar menos consequências em outros sistemas que estejam interligados.

2.3 Web Service

Web Service, ou serviço *Web*, é uma implementação de SOA. É a forma pela qual a arquitetura implementa os serviços (funcionalidades) que são disponibilizados e acessados pelos aplicativos. Muitas vezes há certa confusão entre os termos SOA

e *Web Service*, que na verdade são conceitos e termos distintos, conforme descrito por Josuttis (2008, p.20) “[...] SOA é um paradigma; *Web Services* são uma maneira possível para fornecer a infra-estrutura com o uso de uma estratégia de implementação específica.”

O *Web Service* é a implementação de um serviço SOA para aplicações *Web*, serviços que são disponibilizados em um servidor web para serem acessados através da internet ou intranet da empresa. O *Web Service* apenas implementa as funcionalidades, mas não se preocupa com a *interface* com o usuário final, ficando esta parte para ser implementada por quem for requisitar os serviços (acessar as funcionalidades) do sistema, conforme Pamplona (2008).

Erl (2004) entende que os serviços são como componentes, ou seja, são blocos independentes que coletivamente representam um ambiente da aplicação. De uma forma não tradicional, os serviços têm um número único de características que permitem a eles participar do sistema como parte de SOA. Esta visão do projeto resulta na criação de uma unidade isolada das funcionalidades de negócio que estão em conformidade com um *framework*[1] padrão de comunicação. Devido à independência que os serviços tem com o *framework*, a programação lógica que é encapsulada, não precisa ser restrita a nenhuma plataforma ou tecnologia (ERL, 2004).

Pamplona (2008) entende também que “*Web Services* é a tecnologia ideal para comunicação entre sistemas, sendo muito usado em aplicações B2B [...]”, ou seja, o uso de *Web Services* está cada vez mais presente na comunicação de aplicações B2B (*busines-to-busines*, ou empresa para empresa), pois facilita a integração entre aplicativos de empresas diferentes.

Segundo Cantu (2003, p. 709):

> A tecnologia rapidamente emergente de serviços *Web* tem o potencial de mudar o modo como a *Internet* trabalha a favor das empresas. [...] Os serviços *Web* prestam-se para o intercâmbio de computadores tanto quanto a *Web* e o *e-mail* permitem às pessoas interagirem.

Josuttis (2008, p. 182), define *Web Services* como sendo:

> *Web Services* se referem a um conjunto de padrões que cobrem a interoperabilidade. Na realidade, esses padrões definem tanto os protocolos que são usados na comunicação quanto o formato das interfaces que são usadas para especificar os serviços e contratos de serviços.

[1] Conjunto de códigos-fonte, técnicas e metodologias de desenvolvimento.

Já Nascimento (2007, p.16), conceitua da seguinte forma:

> [...] a solução utilizada na integração de sistemas e na comunicação entre diferentes aplicações, é também uma aplicação que aceita solicitações de outros sistemas através da *Internet*. E ainda, são interfaces acessíveis de rede, para as funcionalidades da aplicação, que utilizam em sua construção tecnologias baseadas nos padrões da *Internet*.

Nascimento (2007) ainda complementa o conceito, onde *Web Service* é a forma utilizada para tornar disponíveis as funcionalidades de um sistema aos usuários *Web*, sejam eles pessoas ou outras aplicações, elencando três fundamentos:

a) viabilizar a convivência entre diferentes aplicativos numa mesma empresa (aplicações distribuídas): integração destes sistemas;
b) compartilhar informações entre diversas entidades (empresas);
c) solucionar problemas na integração de aplicações quando utilizadas técnicas de EAI (*Enterprise Application Integration* – integração de aplicações empresariais) e EDI (*Eletronic Data Interchage* – intercâmbio eletrônico de dados);

A seguir, na Figura 3, é ilustrado a organização da estrutura de um *Web Service*, onde podemos identificar a comunicação 1, entre o UDDI (veremos a seguir o significado) e o provedor (*service provider*), que é o responsável pela publicação da descrição do serviço. Na interação 2, entre o *Web Service* e o UDDI, ocorre a procura pelos serviços. Já nas comunicações 3 e 4 ocorrem as trocas de mensagem entre si, onde o *Web Service* efetua a chamada ao provedor (3) e então o *Web Service* provê o serviço aos usuários (4), que normalmente são aplicações clientes que fazem uso dos serviços.

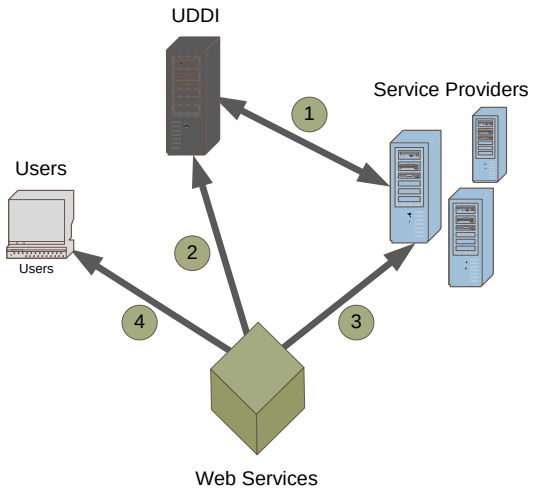


Figura 3 – Organização da estrutura de um Web Service
Adaptado de Nascimento (2007)

Alguns termos que são comuns no uso de *Web Services* necessitam ser apresentados, como: *fornecedor* e *consumidor*.

a) *fornecedor*, é entendido como sendo o sistema ou aplicativo que irá implementar um serviço, uma funcionalidade de negócio, por exemplo. Ou seja, o *fornecedor* é quem será chamado pelos outros aplicativos quando da requisição de uma determinada funcionalidade (serviço);
b) *consumidor*, é quem fará uso do serviço disponibilizado. Ou seja, é quem irá fazer as requisições ao *fornecedor* de uma determinada funcionalidade.

Sendo assim, o termo *consumir um serviço*, significa que uma aplicação ou sistema fará uso (irá requisitar) de um serviço que é *fornecido*. Pode-se fazer uma analogia com os termos *cliente* e *servidor*, onde o *cliente* é o consumidor do serviço e o *fornecedor* é o servidor da aplicação, por exemplo.

Web Services usa alguns padrões em sua abordagem, sendo eles:

a) XML;
b) HTTP (ou HTTPS);
c) WSDL;
d) SOAP;
e) UDDI.

2.3.1 XML – Extensible Markup Language

XML (*Extensible Markup Language* – linguagem de marcação extensível), de acordo com Mello (2003, p. 1), “[...] é um formato padrão para transferência de dados entre computadores [...]” publicado pelo W3C Consortium.

Segundo Cantu (2003, p. 676):

> O XML é uma *linguagem e marcação*, o que significa que ela usa símbolos para descrever seu próprio conteúdo – neste caso, *tags* consistindo em um texto definido de maneira especial, incluído entre os sinais de menor e maior. Ela se chama *extensível* porque permite tags (marcas) livres – em contraste, por exemplo com HTML, que tem tags predefinidas [...].

Mello (2003, p. 1) avança um pouco:

> Do ponto de vista da área de Banco de Dados (BD), um documento XML é uma coleção de dados, da mesma forma que um BD. Porém, um documento XML tem a vantagem de manter dados auto-descritivos e facilmente portáveis. A tecnologia XML assemelha-se bastante com a tecnologia de Sistemas Gerenciadores de BD (SGBDs), uma vez que possui um formato de armazenamento específico (documento), permite a definição de esquemas, possui linguagens de consulta e define interfaces (APIs) para o acesso a dados. Entretanto, a tecnologia XML não é equivalente à tecnologia de SGBD, pois não existem soluções para todos os aspectos de gerenciamento de dados, como controle de integridade, gerenciamento de transações, indexação e consultas a múltiplos documentos.

XML vem sendo aplicado a diversos domínios de aplicação, como comércio eletrônico e cadastro bibliográfico, por exemplo. É um padrão bastante comum a diversas aplicações e vêm sendo utilizado em diversas áreas da ciência da computação, como linguagens de programação, engenharia de *software* e bancos de dados. XML pode ser considerado como o padrão comum para troca de informações pré-formatadas por diversos sistemas.

Josuttis (2008, p. 182) acrescenta que “XML é usado como formato geral para descrever modelos, formatos e tipos de dados [...] todos os padrões de *Web Services* são baseados em XML 1.0, XSD (*XML Schema Definition*) e *namespaces* XML”.

Erl (2004) conceitua XML como uma meta-linguagem que complementa as funcionalidades de apresentação de HTML (*HyperText Markup Language* – linguagem de marcação de hipertexto) com a habilidade para descrever a natureza da informação que está sendo apresentada. Sem XML a informação passada através da Internet tem pouco significado ou contexto, além do valor que é propriamente apresentado.

XML adiciona uma camada de inteligência à informação, proporcional à inteligência que é aplicada para descrever o documento. O padrão XML nos permite adicionar meta-informações que são auto descritivas, ou seja, isto torna cada documento *Web* em um minirrepositório de informações (ERL, 2004).

Erl (2004) entende que o mais bem sucedido e aceito tipo de *Web Service* é o XML *Web Service*, em função deste tipo de serviço ter dois requisitos fundamentais:

a) sua comunicação é feita através de protocolos *Internet* (HTTP é o mais comum);
b) envia e recebe dados formatados como documentos XML.

A troca de informações mais comum entre o fornecedor e o consumidor se dá

por intermédio de documentos XML, seja para obter os dados requisitados, ou para devolver as informações ao fornecedor, quando houver necessidade de retorno.

2.3.2 HTTP – HyperText Transfer Protocol

HTTP (*Hypertext Transfer Protocol* – protocolo de transferência de hipertexto) é o protocolo de comunicação comum utilizado na *Internet* para transferência de informações. HTTP é a forma padronizada de comunicação para distribuir as informações através da internet.

W3C (2009) define HTTP como sendo um protocolo em nível de aplicação para distribuição e colaboração de hipermídia e está em uso na *World-Wide Web* desde 1990. O protocolo também é utilizado como um protocolo genérico para comunicação entre *user agents* (agentes de usuário, programas que acessam um determinado serviço) e diversos sistemas na *Internet*.

Josuttis (2008, p. 182) considera HTTP "[...] um protocolo de baixo nível usado pela *Internet*. HTTP(S) é um protocolo possível que pode ser usado para enviar *Web Services* pelas redes, usando tecnologia de Internet.", ou seja, é o responsável por tratar das requisições e respostas entre um servidor e um cliente, ou no caso de *Web Services*, entre fornecedor e consumidor.

Uma das vantagens no uso do protocolo HTTP está em minimizar os problemas de conexão por bloqueio de *firewall* (dispositivo de segurança de rede), já que é um tráfego comum entre aplicativos *Web*.

2.3.3 WSDL – Web Services Description Language

WSDL (*Web Services Description Language* – linguagem de descrição de serviços para *Web*), como a própria tradução diz, é a linguagem utilizada para descrever os *Web Services*.

Segundo Cantu (2003, p. 710) "Os documentos WSDL são outro tipo de documento XML que fornece a definição de metadados de uma requisição SOAP.",

ou seja, é o WSDL quem irá especificar a *interface* fornecida no serviço.

Josuttis (2008, p. 182), descreve WSDL como sendo:

> WSDL é usado para definir as *interfaces* dos serviços. Na realidade, ele pode descrever dois aspectos diferentes de um serviço: sua assinatura (nome e parâmetros) e seus detalhes de ligação e *deploy* (protocolo e localização).

WSDL pode ser conceituado como mais uma implementação baseada em XML, é um padrão que fornece uma linguagem para descrever a *interface* dos *Web Services* (ERL, 2004).

Conforme Erl (2004), *Web Services* necessitam ser definidos de uma maneira consistente para que eles possam ser descobertos e interligados com outros serviços e aplicações. WSDL serve justamente para isto, é uma especificação pelo W3C Consortium e possibilita que a grande maioria das linguagens possam acessar os serviços através da descrição das definições do *Web Service*.

Erl (2004) especifica que a camada de integração que é introduzida no *Web Service* estabelece um padrão universal reconhecido e suportado pelas interfaces. Como ilustrado na Figura 4, WSDL possibilita a comunicação entre essas camadas, através da padronização nos pontos de descrição. Como podemos verificar na Figura 4, a camada de integração é a representação do WSDL que provê a descrição dos *Web Services*, possibilitando a integração entre diferentes aplicações.

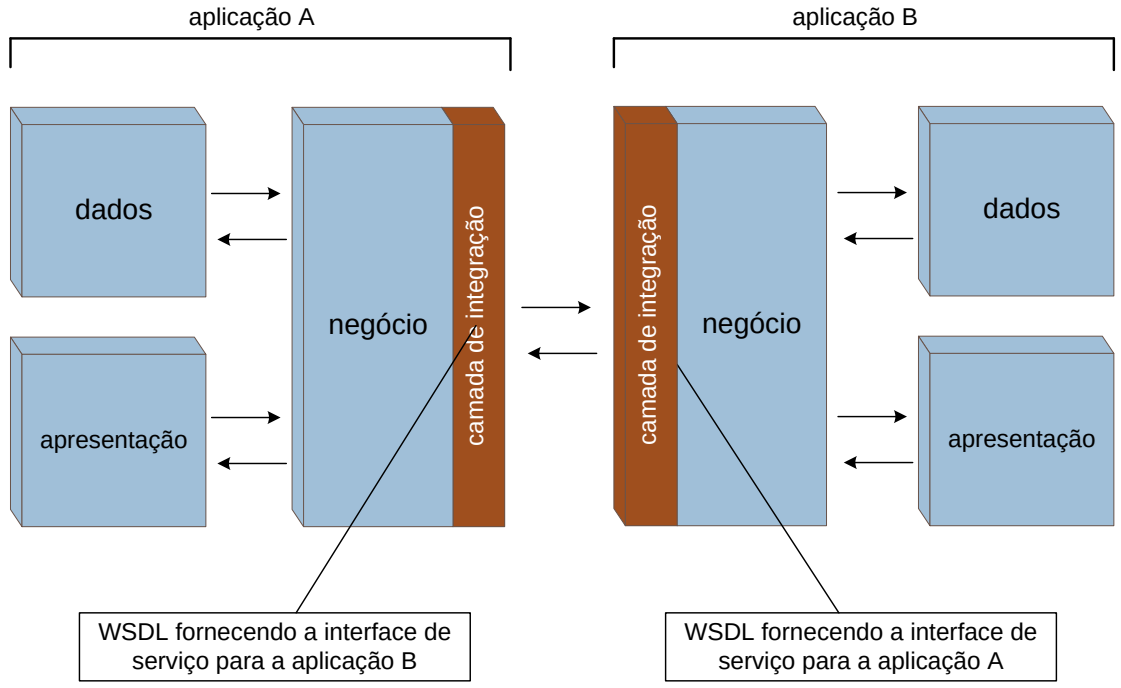


Figura 4 – Representação do documento WSDL fornecendo interface
Adaptado de Erl (2004)

Na Figura 5 pode-se visualizar a estrutura de um arquivo WSDL, nas duas versões disponíveis, onde os documentos WSDL descrevem os *Web Services* iniciando a partir dos tipos de dados e terminando com o endereço do serviço, possuindo três camadas:

a) a primeira camada serve para descrever o serviço. Como ilustrado em "tipo de porta" na Figura 5 (WSDL 1.1) e "*interface*" (WSDL 2.0), onde existem as operações de entradas e saídas como parâmetros específicos. Na versão 1.1, há a necessidade da sessão "mensagem" para definição dos parâmetros, já em 2.0 os parâmetros são definidos como qualquer outro tipo, na seção apropriada;
b) a segunda camada define a forma e o protocolo utilizado na ligação do *Web Service*;
c) a terceira camada fornece a localização do *Web Service*, ou seja, o endereço onde está localizado.

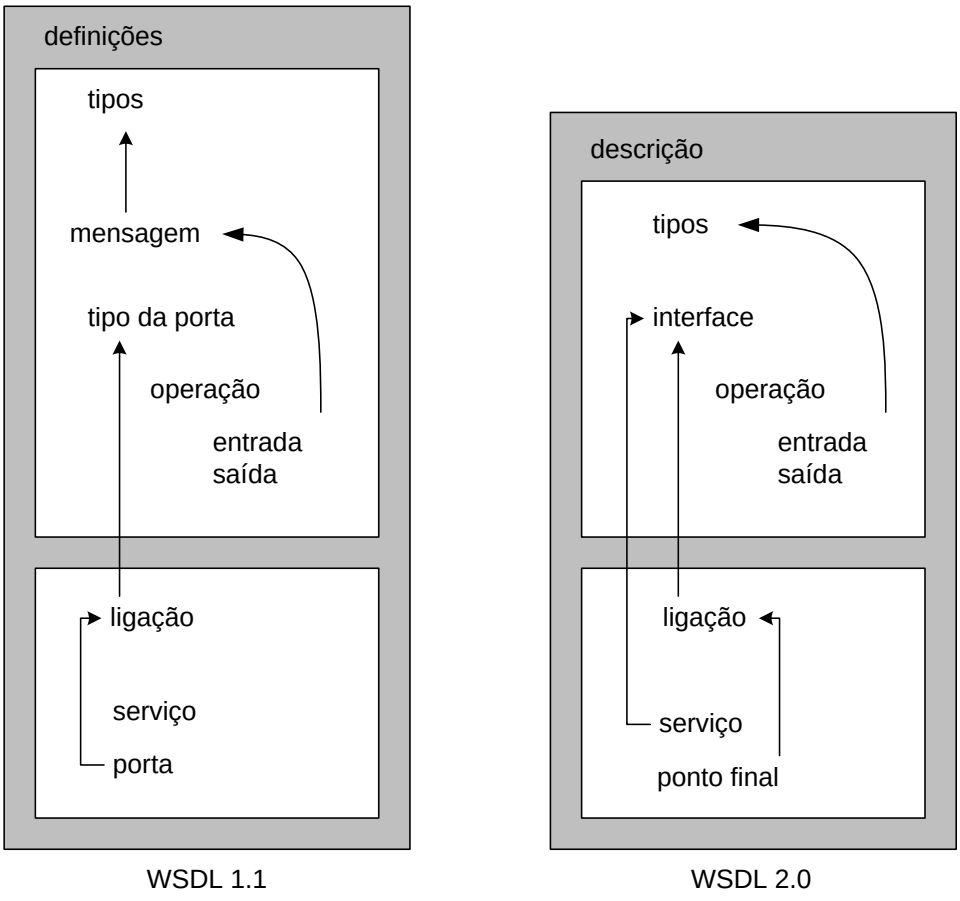


Figura 5 – Estrutura geral de arquivos WSDL 1.1 e 2.0
Adaptado de Josuttis (2008, p. 183)

2.3.4 SOAP – Simple Object Access Protocol

SOAP (*Simple Object Access Protocol* – protocolo simples de acesso a objetos) segundo sua tradução, é um dos protocolos que possibilita o uso de *Web Services* e tem como base o protocolo HTTP para que o servidor Web possa interpretar as requisições aos *Web Services* de forma transparente, sem que haja bloqueio dos pacotes em *firewalls*.

Josuttis (2008, p.187) afirma que “SOAP foi o primeiro padrão real de *Web Services* que foi desenvolvido.”

Cantu (2003, p. 710) apresenta que “SOAP define uma notação baseada em XML para solicitar a execução de um método por um objeto no servidor, passando parâmetros para ele, e uma notação para definir o formato de uma resposta.”

Um complemento para a definição anterior é apresentada por Erl (2004), segundo ele, SOAP é um formato de mensagem baseado em XML, que estabelece um *framework* para comunicação inter-aplicação (ou inter-serviço) por intermédio do protocolo HTTP. Ou seja, a especificação de SOAP estabelece um formato de mensagem padrão que consiste em um documento XML capaz de hospedar RPC (*Remote Procedure Call* – chamada de procedimento remoto) e *document-centric data* (dados centrados no documento) assim como modelo de troca de dados assíncrona.

A mensagem trafegada no protocolo SOAP também é chamada de envelope SOAP, que nada mais é do que um documento no formado XML, conforme mencionado anteriormente e ilustrado nos Quadros 1 e 2, em um exemplo de requisição e outro de resposta.

```
POST /InStock HTTP/1.1
Host: www.example.org
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPrice>
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

Quadro 1 – Exemplo de um envelope de requisição SOAP
Fonte: W3Schools (2009)

```
HTTP/1.1 200 OK
Content-Type: application/soap+xml; charset=utf-8
Content-Length: nnn

<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

  <soap:Body xmlns:m="http://www.example.org/stock">
    <m:GetStockPriceResponse>
      <m:Price>34.5</m:Price>
    </m:GetStockPriceResponse>
  </soap:Body>
</soap:Envelope>
```

Quadro 2 – Exemplo de um envelope de resposta SOAP
Fonte: W3Schools (2009)

Segundo a W3School, a mensagem SOAP contém os seguintes elementos:

a) um elemento *Envelope* que identifica o documento XML como uma mensagem SOAP;

b) um elemento *Header* que contém o cabeçalho da informação;

c) um elemento *Body* que contém a informação de chamada e resposta;

d) um elemento *Fault* que contém as informações de erro e status.

Neste caso, um exemplo do esqueleto de uma mensagem SOAP está ilustrado no Quadro 3:

```
<?xml version="1.0"?>
<soap:Envelope
xmlns:soap="http://www.w3.org/2001/12/soap-envelope"
soap:encodingStyle="http://www.w3.org/2001/12/soap-encoding">

<soap:Header>
  ...
  ...
</soap:Header>

<soap:Body>
  ...
  ...
  <soap:Fault>
    ...
    ...
  </soap:Fault>
</soap:Body>

</soap:Envelope>
```

Quadro 3 – Exemplo de mensagem SOAP
Fonte: W3Schools (2009)

2.3.5 UDDI – Universal Description, Discovery and Integration

UDDI (*Universal Description, Discovery and Integration* – descrição universal, descoberta e integração) é um protocolo que especifica um método para a descrição, publicação e descoberta de *Web Services* em uma arquitetura orientada a serviços, tendo como principal intenção registrar e localizar *Web Services*.

Segundo Josuttis (2008, p. 189):

> A ideia original era introduzir todos os três papéis de um mercado em funcionamento: fornecedores que oferecem serviços, consumidores que precisam dos serviços e *brokers* que juntam os dois divulgando e localizando os serviços.

Erl (2004) afirma, em outras palavras, que UDDI é a especificação que provê um mecanismo padrão para o descobrimento dinâmico da descrição do serviço.

Um dos componentes fundamentais de uma arquitetura orientada a serviço é um mecanismo para que a descrição do *Web Service* (WSDL) seja descoberta pelos consumidores. Para isto é necessário um diretório central que hospede as requisições de descrição dos serviços. Ou seja, essa é a principal importância de UDDI em uma arquitetura orientada a serviço (ERL, 2004).

Cantu (2003, p. 724) define UDDI da seguinte forma:

> A especificação UDDI é um esforço para criar um catálogo de serviços Web oferecido pelas empresas de todo o mundo. O objetivo dessa iniciativa é criar uma estrutura aberta, global e independente de plataforma para que as entidades de negócios se encontrem, definam como interagem com a rede *Internet* e compartilhem de um registro global de negócios [...].

Em um ambiente de rede corporativa (*Intranet*) não há a necessidade de registrar o *Web Service* para que seja descoberto na Internet, quando este não for um serviço de domínio público, por exemplo. Nesta situação o consumidor já irá conhecer o endereço onde o serviço está publicado (servidor *Web*), sem a necessidade de descoberta através da *Internet*. No entanto, em situações em que o serviço seja de domínio público, é interessante registrar a descrição do *Web Service* para que seja facilmente descoberto, inclusive para fins de divulgação do que é oferecido pela empresa ao mercado em termos de tecnologia.

O UDDI define modelos específicos para uma entidade de negócios, para serviço de negócio e modelo de ligação. O tipo de definição de entidade de negócios, inclui informações da empresa, como seus dados cadastrais e categoria de negócio, funcionando similarmente ao conceito de páginas amarelas. A definição dos serviços do negócio relacionam-se às descrições do *Web Service*, onde cada serviço tem um tipo associado, facilitando a busca pelo que está sendo fornecido. Já a definição do tipo de ligação, identifica a tecnologia que está sendo empregada no *Web Service*, seu formato, protocolo e formato do WSDL (CANTU, 2003).

2.4 Banco de dados

Silberschatz, Korth e Sudarshan (1999, p. 1) apresentam a seguinte definição para sistema de banco de dados:

> Um Sistema Gerenciador de Banco de Dados (SGBD) é constituído por um conjunto de dados associados a um conjunto de programas para acesso a esses dados.O conjunto de dados, comumente chamado de banco de dados, contém informações sobre a empresa em particular.

O SGDB, de uma forma simplista, é um *software* com recursos específicos para facilitar a manipulação das informações do banco de dados e o desenvolvimento de programas aplicativos que façam uso das informações. O SGBD é o módulo de programa que fornece a *interface* entre os dados armazenados

num repositório de dados e os programas, ou solicitações submetidas ao sistema.

O principal objetivo dos SGBDs é proporcionar uma forma fácil e rápida para a leitura e gravação das informações, sendo que eles são desenvolvidos para que grandes volumes de informação possam ser gerenciadas, implicando em estruturas próprias para o armazenamento e mecanismos de busca e manipulação das informações contidas no banco de dados (SILBERSCHATZ, KORTH e SUDARSHAN, 1999).

Sendo assim, podemos considerar um banco de dados como uma coleção de dados inter-relacionados que representem informações relativas a um domínio específico.

Silberschatz, Korth e Sudarshan (1999, p. 1) entendem que “[...] um sistema de banco de dados deve garantir a segurança das informações armazenadas contra eventuais problemas com o sistema, além de impedir tentativas de acesso não autorizadas”, ou seja, o SGBD deve fornecer mecanismos que garantam a segurança da informação, para que ela possa ser acessada somente por quem tiver direitos sobre ela.

A forma como as informações armazenadas são acessadas pelos usuários e aplicações ocorrem por intermédio de instruções SQL (*Structured Query Language* – linguagem de consulta estruturada), que é uma linguagem de pesquisa declarativa e que especifica a forma do resultado a ser localizado.

Silberschatz, Korth e Sudarshan (1999, p. 109) afirmam que:

> Embora nos refiramos à linguagem SQL como uma “linguagem de consulta”, ela possui muitos outros recursos além da consulta ao banco de dados, como meios para a definição da estrutura de dados, para modificação de dados no banco de dados e para a especificação de restrições de segurança.

Em outras palavras, SQL é a linguagem padrão de comunicação com a base de dados, é através dela que é realizada toda a comunicação com os dados, seja para consulta, inclusão, exclusão ou modificação dos dados e estrutura, assim como para a restrição de acesso às informações.

3 METODOLOGIA

De acordo com Rezende (2005, p. 105):

> Metodologia não é uma técnica tão-somente, pois se pode utilizar qualquer técnica para o desenvolvimento de projeto, sistema ou *software*, de acordo com a preferência e competência da equipe multidisciplinar envolvida. [...] Desse modo, a metodologia é um roteiro que permite o uso de uma ou várias técnicas por opção dos desenvolvedores do sistema de informação ou *software*.

De outra forma, pode-se entender que a metodologia trata de um conjunto de fatores responsáveis pela estrutura e organização do sistema desenvolvido, como o paradigma de programação utilizado, a arquitetura de *software* empregada, assim como, o modelo de processo de desenvolvimento utilizado no objeto deste trabalho.

A metodologia compreende, também, a forma que foi utilizada para a gerência, análise e o desenvolvimento do aplicativo. Para o presente trabalho de conclusão optou-se por utilizar a Programação Orientada a Objetos como paradigma de programação. Para o modelo de processo de *software*, foram utilizados dois modelos simultaneamente, o Desenvolvimento Incremental e RAD, conforme descrito em item específico a seguir.

Quanto à arquitetura do *software* foi empregada a Arquitetura Orientada a Serviço, conforme detalhado anteriormente na subseção "*SOA – Service-Oriented Architecture*".

3.1 Programação orientada a objetos

Koscianski e Soares (2007, p. 280) entendem que:

> Os diferentes paradigmas de programação correspondem a diversas maneiras de pensar a respeito dos problemas. Podem representar também diferentes algoritmos e, por causa disso, graus variados de eficiência.

O paradigma de programação escolhido para o desenvolvimento do sistema foi a Programação Orientada a Objetos por permitir que o sistema possa ser particionado em diferentes áreas, ou seja, poder modularizar o sistema em partes distintas, além de permitir a reutilização do seu código por intermédio de suas classes.

Sommerville (2003, p. 221), define programação orientada a objetos da seguinte forma:

> A programação orientada a objetos se ocupa de realizar um projeto de *software* utilizando uma linguagem de programação orientada a objetos, por exemplo, a linguagem Java, que aceita a implementação direta de objetos e fornece recursos para definir as classes de objetos.

De acordo com Koscianski e Soares (2007), os objetos representam unidades de um programa e contemplam principalmente os itens a seguir:

a) contêm estados (propriedades) que podem ser invisíveis ao seu exterior;
b) possuem métodos (funções ou ações) que se encarregam de mudar os estados;
c) possibilitam que os seus métodos e estados sejam expostos;
d) são estruturados em hierarquia de classes.

Sommerville (2003, p. 222) tem um entendimento similar sobre o conceito de objeto:

> [...] os termos ‘objeto’ e ‘orientado a objeto’ são amplamente utilizados e aplicados a diferentes tipos de entidades, métodos de projeto, sistemas e linguagens de programação [...] Um objeto é uma entidade que possui um estado e um conjunto definido de operações que operam esse estado. O estado é representado por um conjunto de atributos do objeto [...] Os objetos são criados de acordo com uma definição de classe de objetos, que serve como um *template* para criar objetos.

Os objetos são instanciados (criados) a partir de classes codificadas e estas representam um conjunto de objetos que fazem uso das mesmas características de atributos (propriedades ou estados), operações e semântica (KOSCIANSKI e SOARES, 2007).

Sommerville (2003) complementa ainda que, os sistemas orientados a objeto são de fácil manutenção, visto que os objetos são independentes entre si. Assim, os objetos são entendidos e modificados, quando da necessidade, como entidades *stand-alone* (independentes), sendo assim, a modificação de um objeto ou inclusão de novas funcionalidades podem ocorrer sem que haja interferências em outras partes do sistema (outros objetos).

3.2 Modelos de processo de software

Pressman (2006, p. 16) descreve processo de *software*, como sendo:

> [...] O processo é um diálogo no qual o conhecimento, que deve se transformar em *software* é reunido e incorporado ao software. O processo fornece interação entre usuários e projetistas, entre usuários e ferramentas em desenvolvimento e entre projetistas e ferramentas em desenvolvimento [tecnologia]. É um processo iterativo no qual a própria ferramenta serve como meio de comunicação, com cada nova rodada de diálogo explicitando mais conhecimento útil do pessoal envolvido.

Sommerville (2003, p. 36) define de uma maneira diferente, afirmando que:

> [...] um processo de *software* é um conjunto de atividades e resultados que levam a produção de um produto de software. [...] um modelo de processo de software é uma representação abstrata de um processo de software. Cada modelo de processo representa um processo a partir de uma perspectiva particular, de uma maneira que proporciona apenas informações parciais sobre o processo.

Segundo os autores citados anteriormente, o processo de *software* é considerado como um conjunto de atividades iterativas realizadas durante o desenvolvimento do sistema, desde a sua concepção até a entrega e manutenção, sendo que estas atividades e resultados podem ser adaptados para se ajustar ao aplicativo em questão e à equipe envolvida no processo, procurando aproximar a equipe de desenvolvimento dos usuários do sistema.

Pressman (2006) acrescenta que um modelo de processo de *software* é caracterizado por uma estratégia de desenvolvimento abrangendo camadas de processo, os métodos e ferramentas que auxiliem a equipe durante o processo, e que o desenvolvimento de *software* abrange um ciclo de solução de problema que envolve: a situação atual, definição do problema, desenvolvimento técnico e integração da solução.

O sistema desenvolvido fez uso de partes do modelo incremental e do modelo RAD (*Rapid Application Development* – desenvolvimento rápido de aplicativo), que são descritos a seguir.

3.2.1 Modelo incremental

O modelo incremental pode ser considerado uma evolução do modelo sequencial linear (cascata), pois cada sequência linear (análise, projeto, codificação e testes) gera um incremento do *software*, conforme ilustrado na Figura 6. Desta maneira, cada parte do sistema é considerada como um incremento e assim gera uma parte validada e disponível para uso e, no momento em que todos os incrementos são concluídos, tem-se o produto (*software*) completo e finalizado.

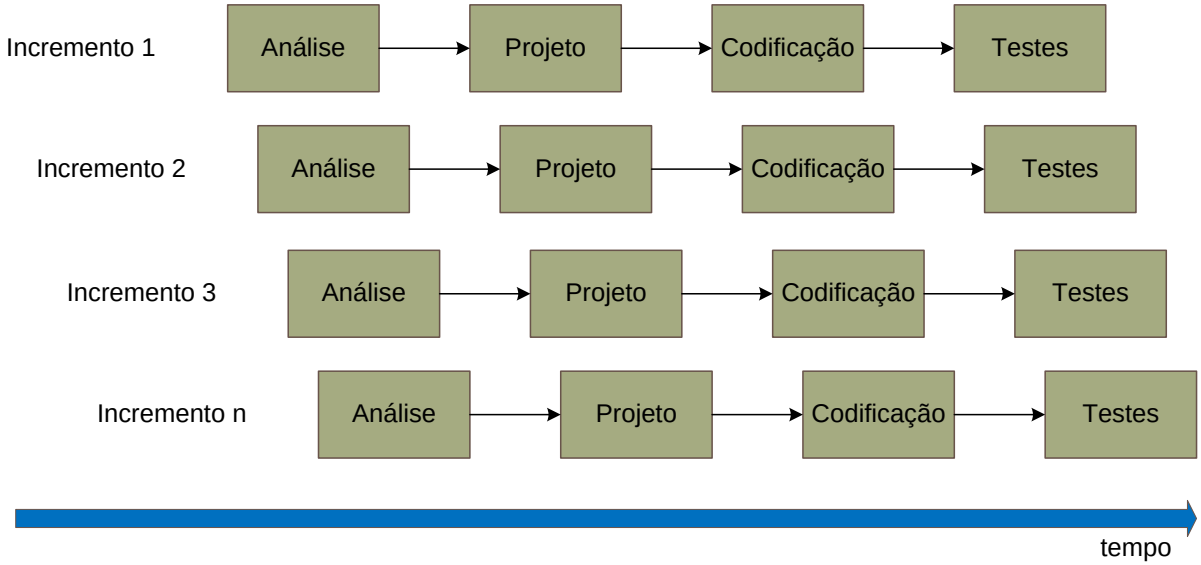


Figura 6 – Fases do desenvolvimento incremental

Pressman (2006, p. 40) entende que

> O modelo incremental combina elementos do modelo em cascata aplicado de maneira iterativa. [...] Cada sequência linear produz “incrementos” de *software* passíveis de serem entregues. [...] Quando um modelo incremental é usado, o primeiro incremento é frequentemente chamado de *núcleo do produto*. Isto é, os requisitos básicos são satisfeitos, mas muitas características suplementares [...] deixam de ser elaboradas.

Sommerville (2003) apresenta que no processo de desenvolvimento incremental, os clientes identificam as funcionalidades fornecidas pelo sistema, assim como o seu grau de importância, para que sejam definidos os estágios de entrega e, dentro destes, um subconjunto de novas funcionalidades.

Para Sommerville (2003, p. 44):

> Uma vez identificados os incrementos, os requisitos para as funções a serem entregues no primeiro incremento são definidos em detalhes, e esse incremento é desenvolvido, utilizando-se o processo de desenvolvimento mais adequado. Durante esse desenvolvimento, outras análises de requisitos para os incrementos seguintes podem ocorrer, mas as mudanças nos requisitos para o incremento atual são aceitas.
> Uma vez que um incremento é concluído e entregue, os clientes podem colocá-lo em operação. Isso significa que eles recebem com antecedência parte da funcionalidade do sistema.

Algumas vantagens da utilização deste modelo (SOMMERVILLE, 2003):

a) não há necessidade que o cliente espere até que todo o *software* esteja desenvolvido para que inicie o seu uso;
b) utilizando o *software* em partes (incrementos), o usuário realiza a validação do sistema de forma gradual;
c) o risco de que o *software* não funcione por completo é mínimo, visto que os problemas podem ser identificados a cada incremento que é entregue;
d) à medida que os incrementos com maior prioridade (grau de importância) são entregues, antes, passam por um volume maior de testes, já que os incrementos posteriores são integrados aos já entregues e, por consequência, são testados novamente.

3.2.2 Modelo RAD

> O *RAD* (*Rapid Application Development, desenvolvimento rápido de aplicação*) é um modelo de processo de *software* incremental que enfatiza um ciclo de desenvolvimento extremamente curto. O modelo RAD é uma adaptação “de alta velocidade” do modelo em cascata, no qual o desenvolvimento rápido é conseguido pelo uso de construção baseada em componentes (PRESSMAN, 2006, p. 41).

O processo RAD é mais adequado para períodos de desenvolvimento curtos, entre 60 a 90 dias e, segundo Pressman (2006), abrange as atividades a seguir:

a) comunicação – envolve a colaboração e comunicação com o cliente, abrangendo levantamento de requisitos. Onde há o entendimento do negócio e as características do *software*;
b) planejamento – estabelece um plano de trabalho, detalhando tarefas e

cronograma; é fundamental para a integração das equipes de desenvolvimento;

c) modelagem – esta atividade esclarece os requisitos e o projeto para que sejam melhor entendidos por parte dos desenvolvedores e clientes. Abrange modelagem do negócio, de dados e dos processos;
d) construção – abrange a codificação e testes do *software*. Faz uso de técnicas de reuso de código e de componentes específicos reutilizáveis para auxiliar na implementação do aplicativo;
e) implantação – o *software* é entregue, então o cliente o avalia e fornece um retorno.

A Figura 7 a seguir, ilustra o processo RAD abortando as áreas descritas anteriormente e enfatizando que cada funcionalidade principal do sistema pode ser atribuída a uma equipe de desenvolvimento distinta.

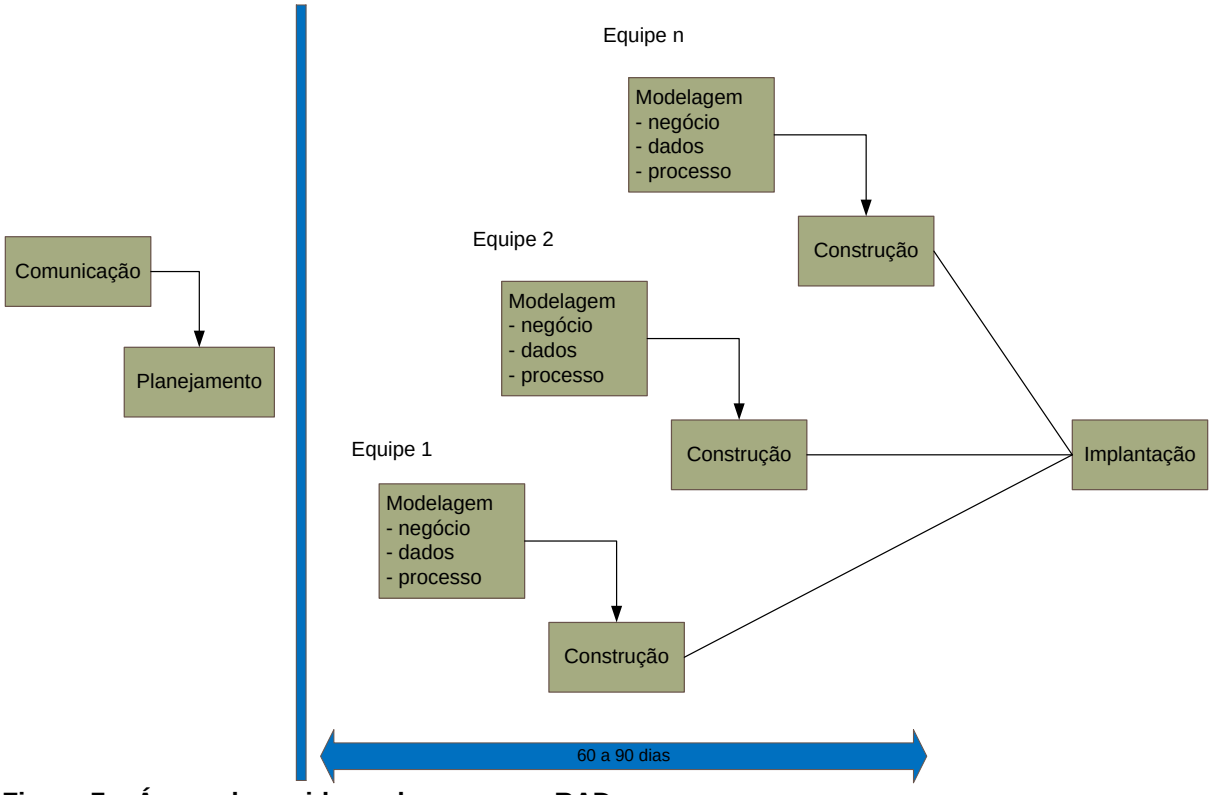


Figura 7 – Áreas abrangidas pelo processo RAD
Adaptado de Pressman (2006)

Como mencionado anteriormente, junto com o modelo incremental também foi utilizado o modelo RAD como processo de *software*, já que também é originado do modelo sequencial linear e tem ênfase em um desenvolvimento rápido, pois faz uso de componentes para a construção do *software.*

A utilização dos dois modelos de processo resulta que cada grande

funcionalidade (módulo), após o levantamento dos requisitos, passará pelos processos de modelagem, implementação (construção ou codificação) e testes. Após cumpridas essas etapas o módulo estará pronto para a sua implantação. Estes módulos, também são considerados como incrementos do *software* e podem ser trabalhados paralelamente, até que se obtenha a finalização de todos os incrementos. Os incrementos foram tratados dentro de uma fase de desenvolvimento curta, como proposto pelo processo RAD, fazendo uso de componentes e reutilização de código para agilizar o tempo de desenvolvimento.

Mesclando os dois processos, foi realizada a subdivisão do projeto em incrementos, onde cada deles passou por fases específicas, sendo elas:

a) análise de requisitos;
b) projeto;
c) desenvolvimento;
d) testes.

As fases de análise de requisitos e projeto são especificadas em subseções específicas a seguir. A fase de desenvolvimento compreende a codificação do sistema de acordo com as tecnologias escolhidas e detalhadas em capítulo específico. Já a etapa de testes, compreende as verificações necessárias a cada um dos incrementos para sua posterior entrega ao cliente.

3.3 Análise de requisitos

Entende-se por requisitos em um *software*, as premissas que necessitam ser atendidas para que o aplicativo tenha êxito quanto às exigências dos usuários, ou seja, é uma declaração do que deve ser atingido quanto aos seus objetivos e aquilo que deve ser feito pelo sistema.

Sommerville (2003, p. 82) descreve requisito como:

> As descrições das funções e das restrições são os *requisitos* para o sistema; e o processo de descobrir, analisar, documentar e verificar essas funções e restrições é chamado de *engenharia de requisitos*. [...] O termo *requisito* não é utilizado pela indústria de *software* de modo consistente. Em alguns casos, um requisito é visto como uma declaração abstrata, de alto nível, de uma função que o sistema deve fornecer ou de uma restrição do sistema.

Koscianski e Soares (2007, p. 174) apresentam uma definição similar à anterior apresentada, segundo eles:

> Os requisitos de um *software* são as descrições sobre seu comportamento, funções e especificações das operações que deve realizar e especificações sobre suas propriedades ou atributos. Os requisitos compreendem as funcionalidades presentes no *software* quando este estiver pronto para ser executado.

Baseado na descrição do problema encontrado durante as entrevistas ocorreu a etapa de análise de requisitos, que é a fase inicial do projeto. Nesta etapa devem ficar bem claras as necessidades dos usuários e os processos que são realizados atualmente, para que estes possam ser melhorados.

3.3.1 Técnica de levantamento de requisitos

Koscianski e Soares (2007, p 174) afirmam que:

> Não existe um processo ideal de levantamento de requisitos (*Requirements Elicitation*) que seja adaptável a todas as empresas. Infelizmente, muitos usuários, desenvolvedores e gerentes não entendem a necessidade de especificações de requisitos e acreditam que é importante começar o mais cedo possível a codificação.

O levantamento de requisitos envolve vários aspectos e formas de abordagem diferente, dependendo de cada situação e para melhor avaliar as necessidades e funcionalidades, foi utilizada a técnica de entrevistas, através de reuniões com possíveis usuários para o aplicativo, neste caso gestores financeiros, que se propuseram a expor as dificuldades enfrentadas, o que facilitou a definição do escopo do sistema.

Um usuário de fundamental importância ao longo do desenvolvimento do sistema foi o prof. Sérgio Nikolay. Além de ser coorientador do presente trabalho de conclusão, também é um usuário do sistema e, em função da sua experiência como consultor financeiro, professor de administração financeira e vice-diretor administrativo financeiro das Faculdades Integradas de Taquara, foi um dos entrevistados na etapa de análise de requisitos.

Outro usuário entrevistado de enorme contribuição foi Roberto Villas Bôas, que atua em empresa da região como analista administrativo financeiro e supervisor de contas a pagar e receber, também foi entrevistado durante o levantamento de

requisitos para definir as funcionalidades necessárias e a abrangência do sistema.

A técnica de entrevistas é uma das técnicas mais comuns, pois o analista reúne várias questões para serem respondidas sobre o sistema e o usuário, ou equipe de pessoas, quando o caso, responde e, quando possível, apresenta informações sobre as funcionalidades necessárias para suprir o problema atual (KOSCIANSKI e SOARES, 2007).

3.3.2 Requisitos funcionais

Os requisitos funcionais descrevem funcionalidades ou serviços que espera-se que um *software* ofereça ao usuário, de acordo com o tipo de sistema que estiver em desenvolvimento. Os requisitos funcionais de usuário devem esclarecer recursos específicos que o sistema deve realizar (SOMMERVILLE, 2003).

Os requisitos funcionais necessitam ser entendidos facilmente pelos usuários, então deve-se utilizar uma linguagem com o mínimo teor técnico possível, para que possa expressar com clareza as necessidades levantadas. De acordo com Koscianski e Soares (2007, p. 179) “requisitos funcionais devem ser desvinculados o máximo possível da tecnologia de implementação”, ou seja, isto confirma que detalhes como linguagem de programação, bibliotecas, componentes ou banco de dados não devem ser mencionados durante a fase de análise de requisitos.

> Em princípio, a especificação de requisitos funcionais de um sistema deve ser completa e consistente. A completeza significa que todas as funções requeridas pelo usuário devem estar definidas. A consistência significa que os requisitos não devem ter definições contraditórias (SOMMERVILLE, 2002, p. 84).

A especificação destes requisitos deve ser completa e consistente, mas para projetos complexos e grandes, muitas vezes torna-se impossível que todas as funcionalidades requeridas pelo usuário sejam especificadas em função de pontos de vista diferentes tanto da equipe de desenvolvedores quanto propriamente de usuários diferentes. Nestes casos o documento de requisitos sofre alterações ao longo do desenvolvimento, a medida que os problemas surgidos são identificados nas fases posteriores do projeto (KOSCIANSKI e SOARES, 2007).

Durante a fase de entrevistas com os usuários foram definidos os seguintes

requisitos funcionais para posterior fase de projeto do aplicativo. Os requisitos foram divididos em requisitos da aplicação cliente e aplicação servidora, conforme segue no Quadro 4.

Aplicação Cliente:

- Classificar os lançamentos em categorias, se possível utilizando estruturação das categorias em árvore.
- Caracterizar os lançamentos em entidades distintas (clientes, fornecedores, etc.)
- Permitir a inclusão de lançamentos de previsão de valores, contemplando as seguintes informações:
    - Entidade (nome do fornecedor, cliente, ...)
    - Categoria de conta (receita ou despesa)
    - Data de emissão
    - Data de previsão (vencimento)
    - Data de confirmação
    - Valor previsto
    - Valor confirmado
    - Número do documento
- Visualizar dados realizados, se possível carregar as informações que foram realizadas no sistema de retaguarda.
- Geração de cubo de decisão, para facilitar a visualização dos lançamentos (previsões) de forma sintética ou analítica em períodos específicos (diário, semanal, mensal, trimestral, anual).
- Analisar as informações em forma de gráficos, com dados baseados no cubo de decisão.
- Possibilidade de impressão de relatórios em períodos específicos (diário, semanal, mensal, trimestral, anual).

Aplicação Servidora:

- Possibilitar o acesso remoto das aplicações cliente.
- Verificar acesso do usuário ao sistema (autenticar usuário).
- Permitir a criação de novos usuários.
- Armazenar os lançamentos de previsões em banco de dados (categorias de contas, entidades, lançamentos, usuários).
- Carregar lançamentos (contas a pagar e receber) do sistema de retaguarda, quando for possível acessar as informações.
- Disponibilizar as informações armazenadas (previsões) e carregadas do sistema (realizadas) para que sejam acessadas pelas aplicações cliente.

Quadro 4 – Descrição dos requisitos

3.3.3 Requisitos não funcionais

Sommerville (2002, p. 85), apresenta a seguinte definição para requisito não funcional:

> Os requisitos não funcionais, como o nome sugere, são aqueles que não dizem respeito diretamente às funções específicas fornecidas pelo sistema. Eles podem estar relacionados a propriedades de sistema emergentes, como confiabilidade, tempo de resposta e espaço em disco. Como alternativa, eles podem definir restrições para o sistema como a capacidade de dispositivos de E/S (entrada / saída) e as representações de dados utilizadas na *interface* do sistema.

Koscianski e Soares (2007, p.179) complementam, “os requisitos não funcionais descrevem restrições ao *software* de forma geral. Não são, portanto, relativos diretamente às funções desempenhadas pelo produto”.

No sistema de fluxo de caixa desenvolvido, foram levantados os seguintes requisitos não funcionais:

a) possibilitar o acesso remoto das informações, não limitando o uso do aplicativo somente à rede local;
b) restringir o acesso aos dados somente aos usuários autenticados no sistema;
c) não limitar o *software* somente a um único sistema gerenciador de banco de dados;
d) possibilitar o acesso às informações contidas em diferentes domínios, desta maneira o usuário poderá escolher a qual base de dados deseja conectar.

3.4 Projeto

Nesta etapa foi realizada a especificação das funcionalidades do aplicativo para que, em conjunto com os usuários, sejam validados e submetidos para aprovação, quanto as suas facilidades de acesso, identificação de campos, fluência e apresentação das informações requisitadas.

Para melhor ilustrar esta etapa, se fez uso da linguagem UML (*Unified*

Modeling Language – linguagem de modelagem unificada) para a modelagem do sistema. Scott (2003, p.19) define da seguinte forma:

> A UML foi projetada para auxiliar aqueles que participam da atividade de *software* a construir modelos que permitam visualizar o sistema, especificar a estrutura e o comportamento deste, construí-lo e documentar as decisões tomadas durante o processo.

Medeiros (2004, p. 10), entende que "[...] UML não nos indica como devemos fazer um *software*. Ela indica apenas as formas que podem ser utilizadas para representar um *software* em diversos estágios de desenvolvimento [...]", ou seja, UML é a forma utilizada para representar graficamente parte do sistema, facilitando o entendimento por parte das pessoas envolvidas na análise e desenvolvimento.

Koscianski e Soares (2007, p. 291) afirmam que:

> O uso de diagramas contribui para que os requisitos sejam mais facilmente compreendidos e documentados. As notações gráficas padronizadas permitem que diferentes profissionais entendam os mesmos conceitos e ideias, evitando-se as ambiguidades existentes em linguagem natural.

Booch, Rumbaugh e Jacobson (2000) entendem que UML é simplesmente uma linguagem, ou seja, é utilizada como parte do processo de desenvolvimento de *software*, pois, é independente do processo escolhido, mas ajusta-se naturalmente aos processos iterativo e incremental. Os autores acrescentam mais, afirmando que a UML é útil para visualização, especificação, construção e documentação do *software*.

É com a finalidade de melhor ilustrar as funcionalidades do sistema que na fase de projeto estão compreendidos os diagramas UML, que é uma linguagem gráfica de modelagem muito bem aceita na indústria de *software*, sendo um padrão mundial para a modelagem das diversas fases de desenvolvimento. A seguir, procura-se esclarecer as especificações das funcionalidades do sistema, através de:

a) casos de uso;
   - diagramas de casos de uso;
   - detalhamento dos casos de uso;
   - diagramas de atividade;
b) diagrama entidade-relacionamento;
c) diagrama de classes.

3.4.1 Casos de uso

Os *casos de uso,* segundo Scott (2003), representam uma sequência de ações, que são executadas por pessoas ou entidades, chamados de *atores*, ou pelo sistema, que irão produzir resultados.

Booch, Rumbaugh e Jacobson (2000, p. 217) definem da seguinte forma:

> Um caso de uso especifica o comportamento de um sistema ou de parte de um sistema e é uma descrição de um conjunto de sequências de ações [...]. Os casos de uso podem ser aplicados para captar o comportamento pretendido pelo sistema que está sendo desenvolvido, sem ser necessário especificar como esse comportamento é implementado.

Sendo assim, os casos de uso oferecem uma forma para que os desenvolvedores e usuários possam trocar informações. A partir dos requisitos levantados, há a necessidade de verificar quais das funcionalidades necessitam ser melhor exploradas através de representações em casos de uso, já que um caso de uso é a representação de um requisito funcional como um todo.

Medeiros (2004) entende que o caso de uso é a parte mais importante do desenvolvimento, pois é o único instrumento que acompanha um *software* do início ao fim, pois é uma ferramenta de consulta, acerto, discussão, reuniões e alterações nas funcionalidades requisitadas.

No projeto do presente sistema, foram usados diagramas de casos de uso, com seus respectivos detalhamentos, para expressar as funcionalidades abrangidas nos requisitos e escopo levantados na etapa de análise.

3.4.1.1 Diagramas de casos de uso

“Os diagramas de casos de uso têm um papel central para a modelagem do comportamento de um sistema. [...] Cada um mostra um conjunto de casos de uso e atores e seus relacionamentos” (BOOCH, RUMBAUGH e JACOBSON, 2000, p. 231).

Nas Figuras 8 e 9 a seguir estão ilustrados os diagramas de casos de uso identificados para a aplicação cliente e para a aplicação servidor do fluxo de caixa

com a finalidade de caracterizar as principais funcionalidades pertinentes a cada uma das aplicações, procurando especificar e documentar o comportamento de cada conjunto.

a) aplicação cliente

Figura 8 – Diagrama de casos de uso da aplicação cliente

b) aplicação servidora

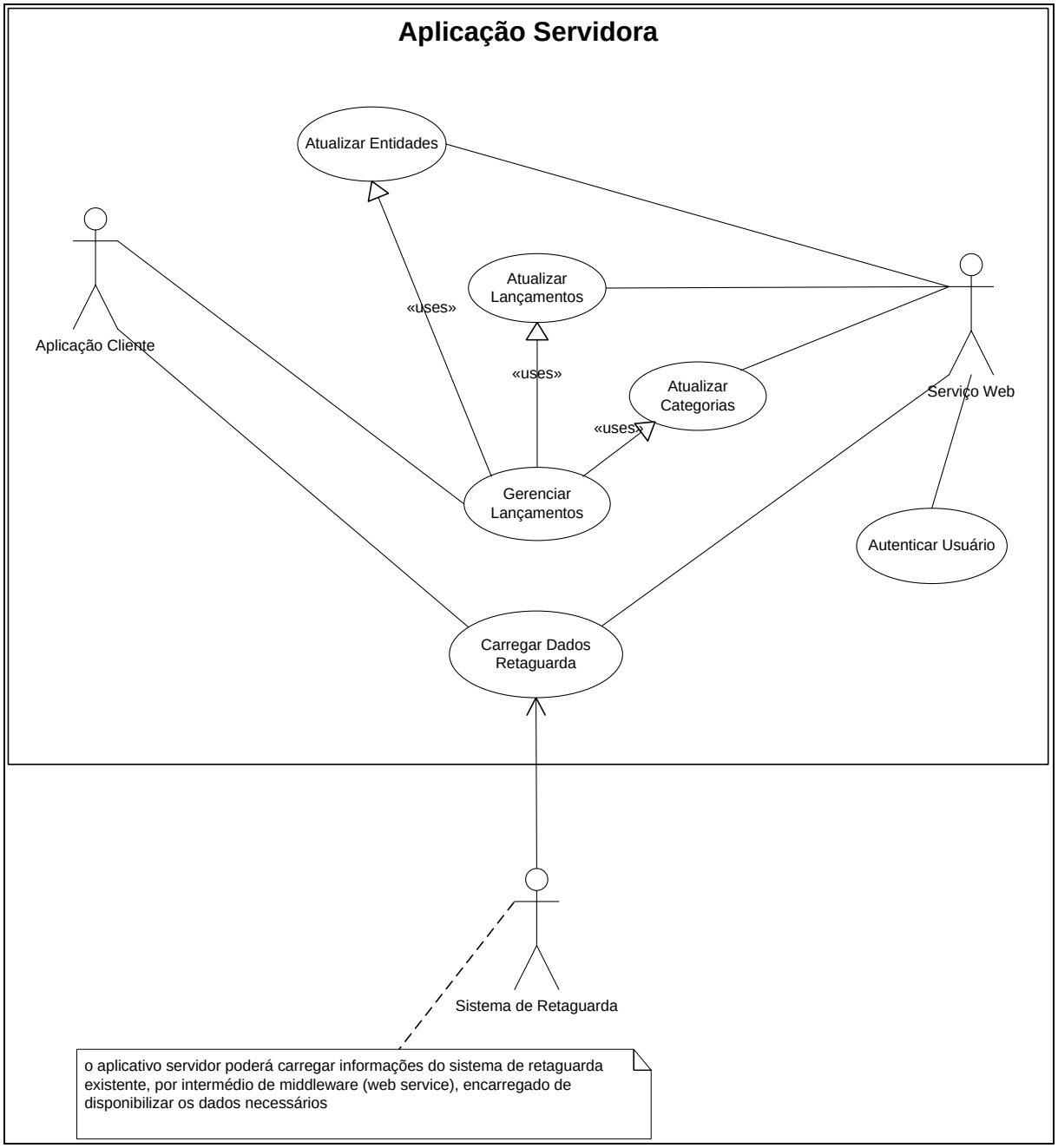


Figura 9 – Diagrama de casos de uso da aplicação servidora

3.4.1.2 Detalhamento dos casos de uso

O detalhamento dos casos de uso são mais importantes do que somente sua visualização em forma de diagramas e por isso são descritos textualmente nos quadros a seguir para facilitar o entendimento dos casos de uso.

a) aplicação cliente

Nome	ucLogin	**Versão**	1.1
Descritivo	Efetuar login no sistema		
Atores envolvidos	Operador, Administrador do Sistema, Serviço Web		
Cenário principal: 1. O sistema deverá restringir o acesso a dois tipos de usuários distintos: administradores e operadores. 2. Os usuários administradores terão acesso irrestrito ao sistema. Os usuários operadores não terão acesso ao cadastro de usuários, categorias e entidades. Eles terão permissão somente para efetuar lançamentos e consultar o fluxo de caixa. 3. O login de acesso ao sistema permitirá que os usuários autenticados acessem o aplicativo cliente e possam operar o sistema. 4. Ao carregar o aplicativo será exibida a tela de login do sistema onde o usuário precisará preencher os dados usuário e senha. 5. Após informar os dados necessários o usuário irá requisitar o login no servidor web, através de serviço web específico, e aguardar a resposta do servidor. 6. Se houver falha na conexão com o servidor, ou a resposta da autenticação for negativa, o aplicativo cliente deve notificar o usuário. 7. Se a validação do usuário e/ou senha for positiva, o aplicativo irá carregar o menu principal do aplicativo cliente.			
Dados: Usuário, alfanumérico de 15 posições Senha, alfanumérico de 10 posições			

Quadro 5 – Detalhamento Efetuar login

Nome	ucCriarUsuario	**Versão**	1.0
Descritivo	Cadastrar usuários para acessar o sistema		
Atores envolvidos	Administrador do Sistema, Serviço Web		

Cenário principal:

1. Os usuários que necessitam acessar o sistema devem ser cadastrados por intermédio do aplicativo cliente.
2. As informações solicitadas para criar um novo usuário são: username, nome, e-mail e uma identificação se o usuário é um administrador do sistema ou não.
3. Cada usuário terá um identificador numérico (código) que será atribuído automaticamente pelo sistema.
4. Além do identificador numérico ele será identificado por um username, que será um nome abreviado e, também irá caracterizar um usuário como único para acesso ao aplicativo.
5. A senha para o novo usuário será atribuída automaticamente a palavra “senha”, que deverá ser alterada quando o usuário realizar login no sistema.
6. Após acessar o cadastro de usuários o administrador poderá inserir novos usuários, remover, alterar ou localizar outros usuários já existentes.
7. Depois de informar os dados requisitados no cadastro, o aplicativo cliente deve realizar uma validação prévia dos dados e campos requeridos e, estando os dados corretos, realizar o registro em cache das operações. Caso as informações estejam incorretas, o administrador deve ser notificado.
8. O aplicativo cliente irá manter mais de uma operação em cache (inclusão, exclusão e alteração), neste caso, quem estiver utilizando o sistema irá decidir se deseja aplicar o conjunto de operações no servidor ou então desfazer.
9. Ao aplicar, as mudanças serão enviadas ao servidor, por intermédio do serviço web, que então irá realizar a gravação e, em caso de falha, será exibida uma notificação do que ocorreu.

Dados:

Identificador, numérico, auto-incrementado
Usuário, alfanumérico de 15 posições
Senha, alfanumérico de 10 posições
Nome, alfanumérico de 50 posições
E-mail, alfanumérico de 255 posições
Administrador, lógico (sim / não)

Quadro 6 – Detalhamento Criar Usuário

Nome	ucCadastrarCategorias	**Versão**	1.1
Descritivo	Cadastrar categorias de lançamentos		
Atores envolvidos	Administrador do Sistema, Serviço Web		

Cenário principal:

1. Os lançamentos que forem registrados serão subdivididos em categorias.
2. As informações solicitadas para criar uma nova categoria são: descrição, tipo de movimento (crédito ou débito), permite lançamento ou não.
3. Cada categoria terá um identificador numérico (código) que será atribuído automaticamente pelo sistema.
4. Além do identificador numérico, cada categoria será caracterizada como única pela sua descrição e tipo de movimentação.
5. Uma categoria poderá estar associada a uma categoria mestre, permitindo que possa ser constituída uma estrutura de árvore de categorias, sendo que elas poderão ser caracterizadas como categorias que permitam ou não que os lançamentos de movimentação possam ser associados.
6. Após acessar o cadastro de categorias o administrador poderá inserir novos registros, remover, alterar ou localizar outros já existentes.
7. Depois de informar os dados requisitados no cadastro, o aplicativo cliente deve realizar uma validação prévia dos dados e campos requeridos e, estando os dados corretos, realizar o registro em cache das operações. Caso as informações estejam incorretas, o administrador deve ser notificado.
8. O aplicativo cliente irá manter mais de uma operação em cache (inclusão, exclusão e alteração), neste caso, quem estiver utilizando o sistema irá decidir se deseja aplicar o conjunto de operações no servidor ou então desfazer.
9. Ao aplicar, as mudanças serão enviadas ao servidor, por intermédio do serviço web, que então irá realizar a gravação e, em caso de falha, será exibida uma notificação do que ocorreu.

Dados:

Identificador, numérico, auto-incrementado
Descrição, alfanumérico de 40 posições
Tipo de movimentação, caractere de 1 posição (Crédito / Débito)
Permite lançamento, lógico (sim / não)
Categoria pai, vínculo para lista de categorias

Quadro 7 – Detalhamento Cadastrar Categorias

Nome	ucCadastrarEntidades	**Versão**	1.0
Descritivo	Cadastrar entidades atribuídas aos lançamentos		
Atores envolvidos	Administrador do Sistema, Serviço Web		

Cenário principal:

1. Os lançamentos que forem registrados serão classificados em entidades (clientes, fornecedores, etc.).
2. As informações solicitadas para criar uma nova entidade são: nome e tipo de entidade.
3. Cada entidade terá um identificador numérico (código) que será atribuído automaticamente pelo sistema.
4. Além do identificador numérico, cada entidade será caracterizada como única pelo seu nome e tipo.
5. Após acessar o cadastro de entidades o administrador poderá inserir novos registros, remover, alterar ou localizar outros já existentes.
6. Depois de informar os dados requisitados no cadastro, o aplicativo cliente deve realizar uma validação prévia dos dados e campos requeridos e, estando os dados corretos, realizar o registro em cache das operações. Caso as informações estejam incorretas, o administrador deve ser notificado.
7. O aplicativo cliente irá manter mais de uma operação em cache (inclusão, exclusão e alteração), neste caso, quem estiver utilizando o sistema irá decidir se deseja aplicar o conjunto de operações no servidor ou então desfazer.
8. Ao aplicar, as mudanças serão enviadas ao servidor, por intermédio do serviço web, que então irá realizar a gravação e, em caso de falha, será exibida uma notificação do que ocorreu.

Dados:

Identificador, numérico, auto-incrementado
Nome, alfanumérico de 50 posições
Tipo de entidade, vínculo para lista de tipos de entidades utilizados

Quadro 8 – Detalhamento Cadastrar Entidades

Nome	ucEfetuarLancamentos	**Versão**	1.1
Descritivo	Realizar os lançamentos de previsão		
Atores envolvidos	Operador, Administrador do Sistema, Serviço Web		

Cenário principal:

1. Os lançamentos de previsão que forem registrados serão subdivididos em categorias e entidades específicos.
2. As informações solicitadas para incluir um novo lançamento são: data de lançamento, data de previsão, data de confirmação, número de documento, valor previsto, valor confirmado, entidade e categoria.
3. Cada lançamento terá um identificador numérico (código) que será atribuído automaticamente pelo sistema.
4. Condições para validação do lançamento:
   4.1. Data de lançamento igual ou superior a data corrente.
   4.2. Data de previsão igual ou superior a data do lançamento.
   4.3. Data de confirmação igual ou superior a data de previsão.
   4.4. Valor previsto deve ser superior a zero (0).
   4.5. Estar associado a uma entidade e categoria pré-cadastradas.
5. Após acessar o controle de lançamento de previsões o operador ou administrador poderá inserir novos registros, remover, alterar ou localizar outros já existentes.
6. Depois de informar os dados requisitados para o lançamento, o aplicativo cliente deve realizar uma validação prévia dos dados e campos requeridos e, estando os dados corretos, realizar o registro em cache das operações. Caso as informações estejam incorretas, o administrador deve ser notificado.
7. O aplicativo cliente irá manter mais de uma operação em cache (inclusão, exclusão e alteração), neste caso, quem estiver utilizando o sistema irá decidir se deseja aplicar o conjunto de operações no servidor ou então desfazer.
8. Ao aplicar, as mudanças serão enviadas ao servidor, por intermédio do serviço web, que então irá realizar a gravação e, em caso de falha, será exibida uma notificação do que ocorreu.

Dados:

Identificador, numérico, auto-incrementado
Data de lançamento, data
Data de previsão, data
Data de confirmação, data
Número de documento, alfanumérico de 15 posições
Entidade, vínculo para lista de entidades
Categoria, vínculo para lista de categorias
Valor previsto, numérico com decimais
Valor confirmado, numérico com decimais

Quadro 9 – Detalhamento Efetuar Lançamentos

Nome	ucImportarDados	**Versão**	1.0
Descritivo	Importar dados de sistema de retaguarda		
Atores envolvidos	Administrador do Sistema, Serviço Web		
Cenário principal: 1. Os dados do sistema de retaguarda (ERP) poderão ser carregados para o fluxo de caixa, por intermédio de arquivos XML de acordo com esquema pré-definido. 2. A importação irá envolver os dados para o cadastro de categorias, entidades e os lançamentos de valores (contas a pagar e receber). 3. O aplicativo cliente deve especificar qual conjunto de dados devem ser carregados (categorias, entidades ou lançamentos). 4. Após identificar o conjunto de dados o administrador do sistema irá requisitar os dados do serviço web específico e então apresentá-los. 5. Os registros são apresentados, já devidamente identificados se são registros novos ou já importados anteriormente. 6. O usuário administrador poderá especificar quais serão os registros a serem importados, marcando-os para isto. 7. Depois de especificados os registros que deseja importar, o usuário irá executar o processo de importação, enviando ao serviço web os registros marcados. 8. O serviço web na aplicação servidora irá realizar a carga dos dados e retornar a aplicação cliente se foi realizado com sucesso ou não.			
Dados: Entidades, Categorias e Lançamentos, conjunto de registros de XML específico			

Quadro 10 – Detalhamento Importar Dados

Nome	ucConsultarFluxo	**Versão**	1.1
Descritivo	Consultar o Fluxo de Caixa		
Atores envolvidos	Operador, Administrador do Sistema, Serviço Web		

Cenário principal:

1. Os lançamentos de previsão que forem registrados, assim como os lançamentos existentes no sistema de retaguarda (quando disponível) irão compor a base para a consulta do Fluxo de Caixa.
2. O fluxo de caixa deve ser apresentado em forma de cubo de decisão, gráficos e relatórios.
3. Os lançamentos devem ser selecionados por um determinado período (intervalo de datas) e agrupados por conjunto de períodos: mensal, trimestral, semestral ou anual.
4. A exibição dos lançamentos deve agrupar os lançamentos em categorias e entidades.
5. O aplicativo cliente deve permitir que os parâmetros a seleção das opções e então requisitar os lançamentos ao servidor por intermédio do serviço web.
6. Condições para requisitar os lançamentos:
   6.1. Período de seleção válido (intervalo de datas).
   6.2. Especificar como deve ser agrupado o conjunto de período.
   6.3. Especificar a forma de apresentação.
7. A consulta poderá ser acessada pelo operador ou administrador do sistema.
8. Depois de recebida a resposta do servidor com o conjunto de dados, apresentá-los de acordo com a forma escolhida.

Dados:

Identificador, numérico, auto-incrementado
Data de lançamento, data
Data de previsão, data
Data de confirmação, data
Número de documento, alfanumérico de 15 posições
Entidade, vínculo para lista de entidades
Categoria, vínculo para lista de categorias
Valor previsto, numérico com decimais
Valor confirmado, numérico com decimais

Quadro 11 – Detalhamento Consultar Fluxo

b) aplicação servidora

Nome	ucAutenticarUsuario	**Versão**	1.0
Descritivo	Autorizar o acesso ao aplicativo (login / logout)		
Atores envolvidos	Aplicativo Cliente e Serviço Web		

Cenário principal:

1. A autenticação do usuário envolverá login e logout do sistema.
2. Deverá verificar quais os usuários que terão permissão de acesso ao aplicativo cliente.
3. O aplicativo cliente irá requisitar ao servidor, por intermédio do serviço web, a autenticação do usuário (login) ou logout do mesmo.
4. O servidor deverá identificar o tipo de requisição (login / logout) e então encaminhar para o procedimento específico.
5. Quando for uma requisição de login, o aplicativo servidor deve verificar os parâmetros informados (username e senha). Caso os parâmetros não forem informados na requisição, a aplicação cliente deve ser notificada.
6. Depois de verificados os parâmetros, o servidor deve consistir na base se existe algum usuário com o username e senha informados. Caso não exista, a aplicação cliente será notificada, do contrário o usuário será registrado em sessão.
7. Se a requisição for de logout, o servidor deverá verificar se o usuário informado está “logado” no sistema.
8. Caso o usuário não esteja registrado em sessão, o servidor deverá notificar a aplicação cliente. Se estiver registrado, então o servidor deverá liberar o usuário registrado.
9. Ao final do processo, caso o login ou logout tenha sucesso, então a requisição deve ser respondida ao aplicativo cliente.

Dados:

Usuário, alfanumérico de 15 posições

Senha, alfanumérico de 10 posições

Quadro 12 – Detalhamento Autenticar Usuário

Nome	ucGerenciarLancamentos	**Versão**	1.2
Descritivo	Requisitar os lançamentos realizados na base de dados		
Atores envolvidos	Aplicativo Cliente e Serviço Web		
Cenário principal: 1. O aplicativo servidor irá prover acesso aos dados registrados no sistema aos usuários autenticados e com permissão de acesso. 2. O servidor deverá verificar se o usuário que está requisitando os lançamentos está autenticado, caso não esteja, irá notificar a aplicação cliente. 3. Se o usuário estiver autenticado, o servidor deverá verificar se os parâmetros da funcionalidade específica foram informados, caso não sejam identificados o aplicativo cliente será notificado. 4. Estando os parâmetros de corretos, a funcionalidade específica será processada e o resultado deverá ser retornado pelo serviço web para o aplicativo cliente no formato XML esperado.			
Dados: Identificador, numérico, auto-incrementado Data de lançamento, data Data de previsão, data Data de confirmaçao, data Número de documento, alfanumérico de 15 posições Entidade, vínculo para lista de entidades Categoria, vínculo para lista de categorias Valor previsto, numérico com decimais Valor confirmado, numérico com decimais			

Quadro 13 – Detalhamento Gerenciar Lançamentos

Nome	ucCarregarDadosRetaguarda	**Versão**	1.2
Descritivo	Carrega os dados do sistema de retaguarda		
Atores envolvidos	Serviço Web		
Cenário principal: 1. O *middleware* de conexão com o sistema de retaguarda será um serviço web específico e carregado pelo aplicativo servidor e não diretamente pelo aplicativo cliente. 2. Devem existir funcionalidades específicas para os módulos de categorias, entidades e lançamentos. 3. Ao carregar o serviço, caso exista configuração para acesso ao sistema de retaguarda, o serviço web irá carregar os dados do módulo especificado e então gerar um arquivo XML dentro do formato especificado. 4. Depois de verificada a existência do arquivo XML específico, tendo sido gerado pelo serviço web ou a partir de serviço de terceiros, o seu conteúdo deve ser validado para verificar se está de acordo com o esquema de dados pré-definido. 5. Caso não exista o arquivo apropriado ou a estrutura esteja incorreta, o servidor será notificado e o processo finalizado. 6. Se estiverem corretos, os registros são carregados e os dados são retornados para a aplicação cliente. 7. A aplicação cliente irá identificar para o serviço web quais registros devem ser salvos na base de dados do sistema de fluxo de caixa. 8. Quando o serviço web receber os registros que devem ser importados, ele irá identificar a qual módulo específico os registros pertencem, e então ser encarregará de armazená-los apropriadamente e retornar caso haja alguma falha.			
Dados: Entidades (clientes ou fornecedores) , conjunto de registros de XML específico Categorias (grupos de contas) , conjunto de registros de XML específico Lançamentos (contas a pagar e receber), conjunto de registros de XML específico			

Quadro 14 – Detalhamento Carregar Dados Retaguarda

3.4.1.3 Diagramas de atividades

Os diagramas de atividades são utilizados para expressar graficamente o fluxo de um caso de uso e visualizar o fluxo de controle de uma atividade para outra. Eles ilustram as funcionalidades de um caso de uso específico.

Medeiros (2004, p. 62) afirma que “O diagrama de atividades existe para ajudá-lo a criar boas descrições de Caso de Uso [...]”.

A seguir estão ilustrados os diagramas de atividade dos casos de uso descritos anteriormente.

a) aplicação cliente

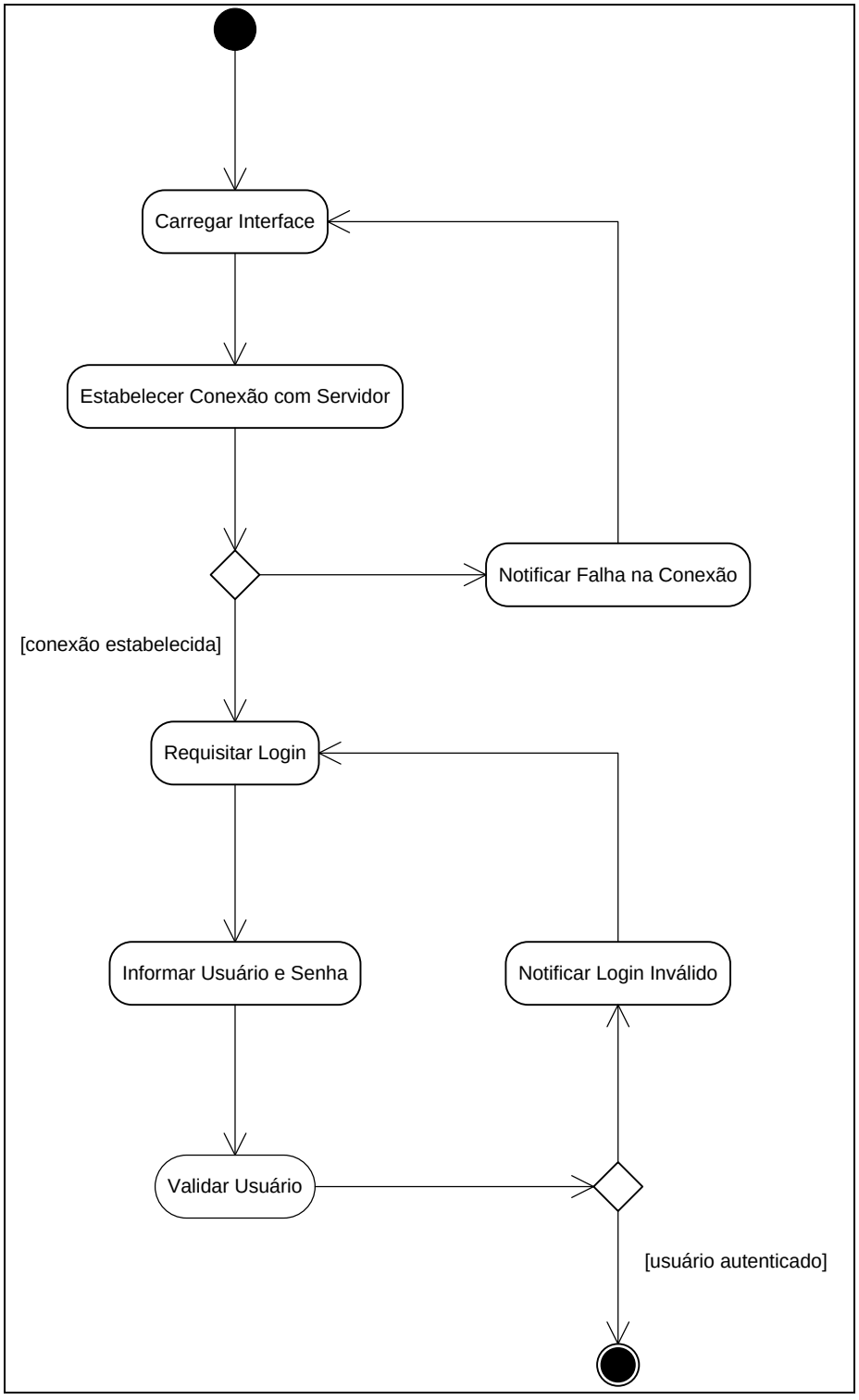


Figura 10 – Diagrama de atividades Efetuar Login

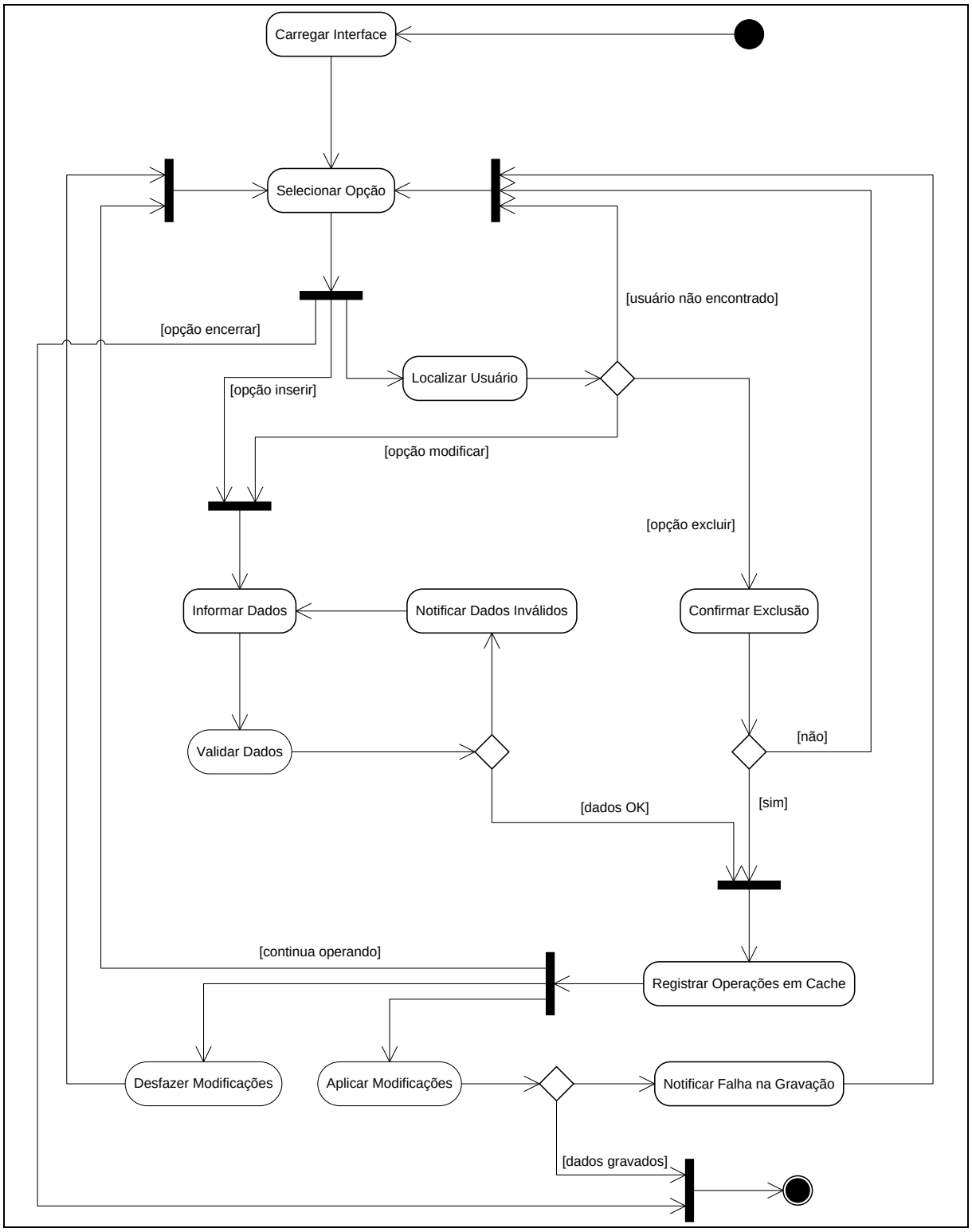


Figura 11 – Diagrama de atividades Cadastrar Usuários

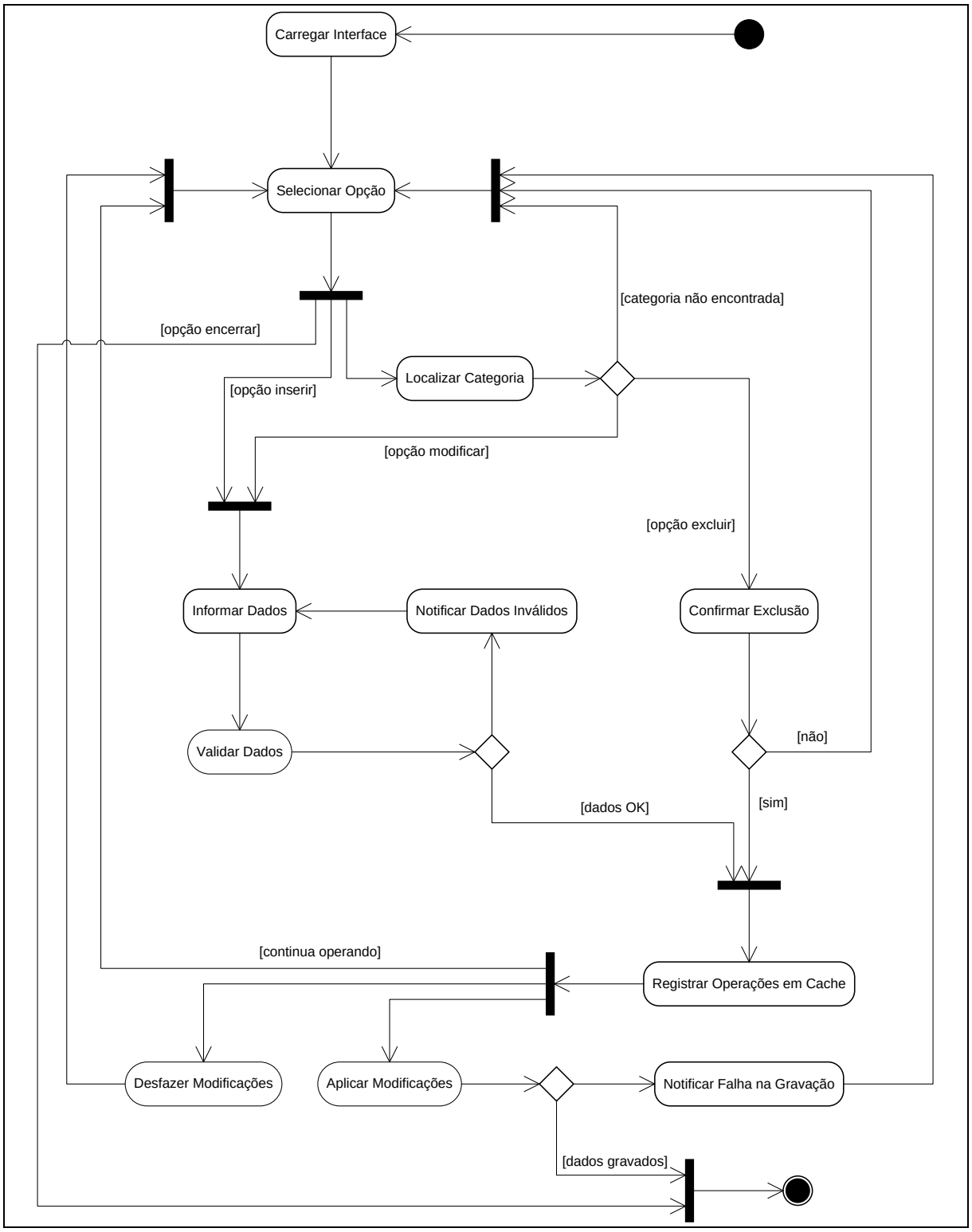


Figura 12 – Diagrama de atividades Cadastrar Categorias

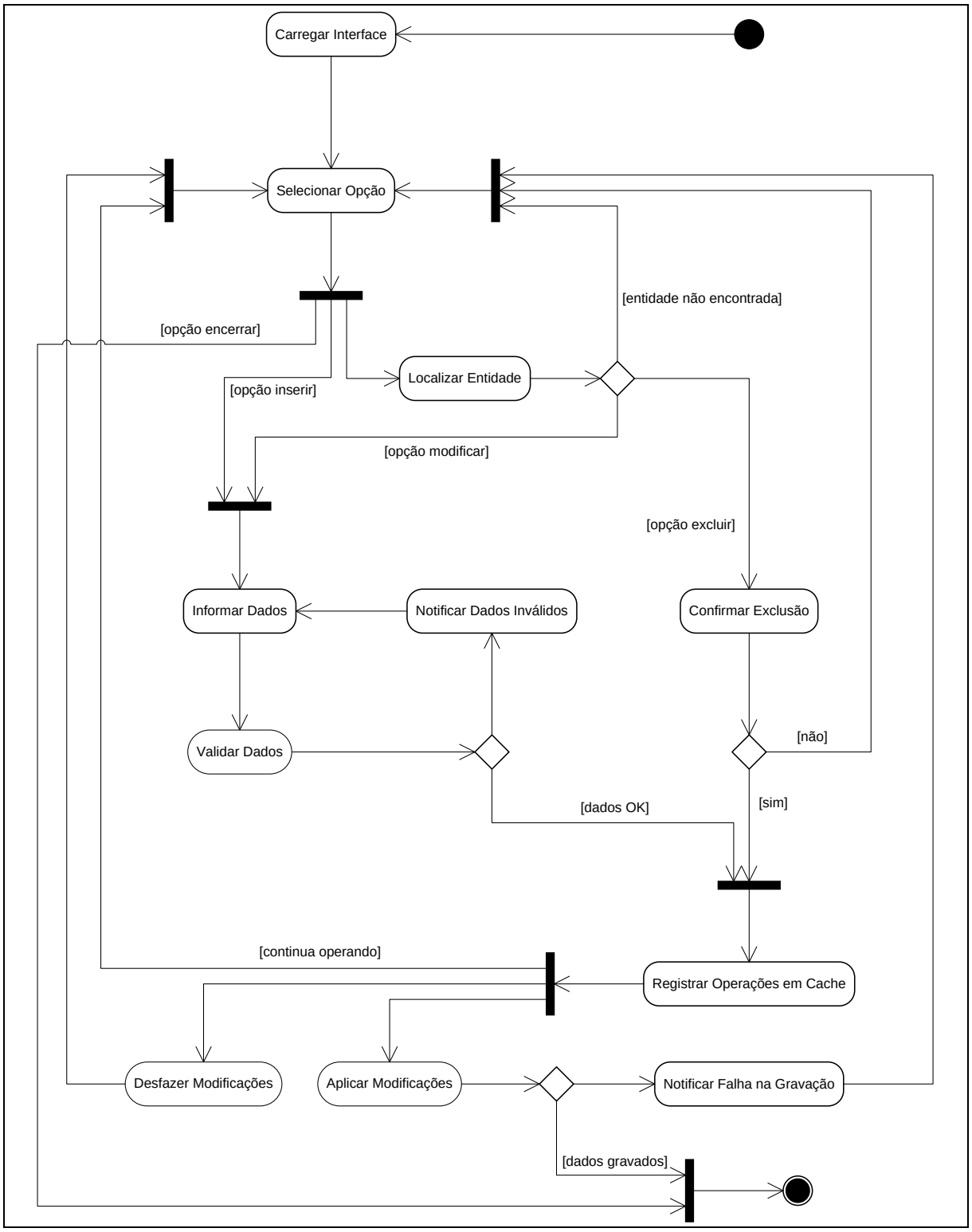


Figura 13 – Diagrama de atividades Cadastrar Entidades

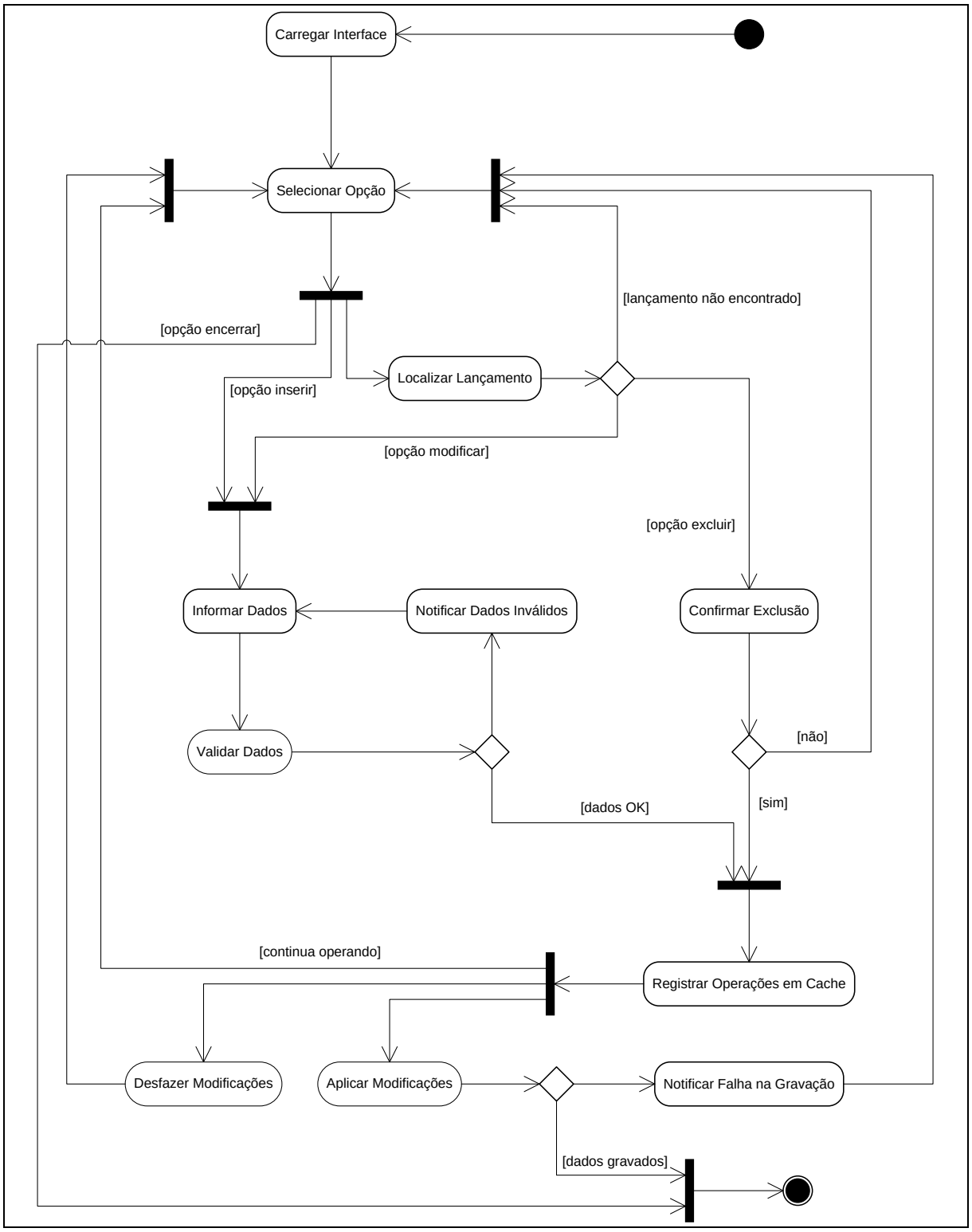


Figura 14 – Diagrama de atividades Efetuar Lançamentos

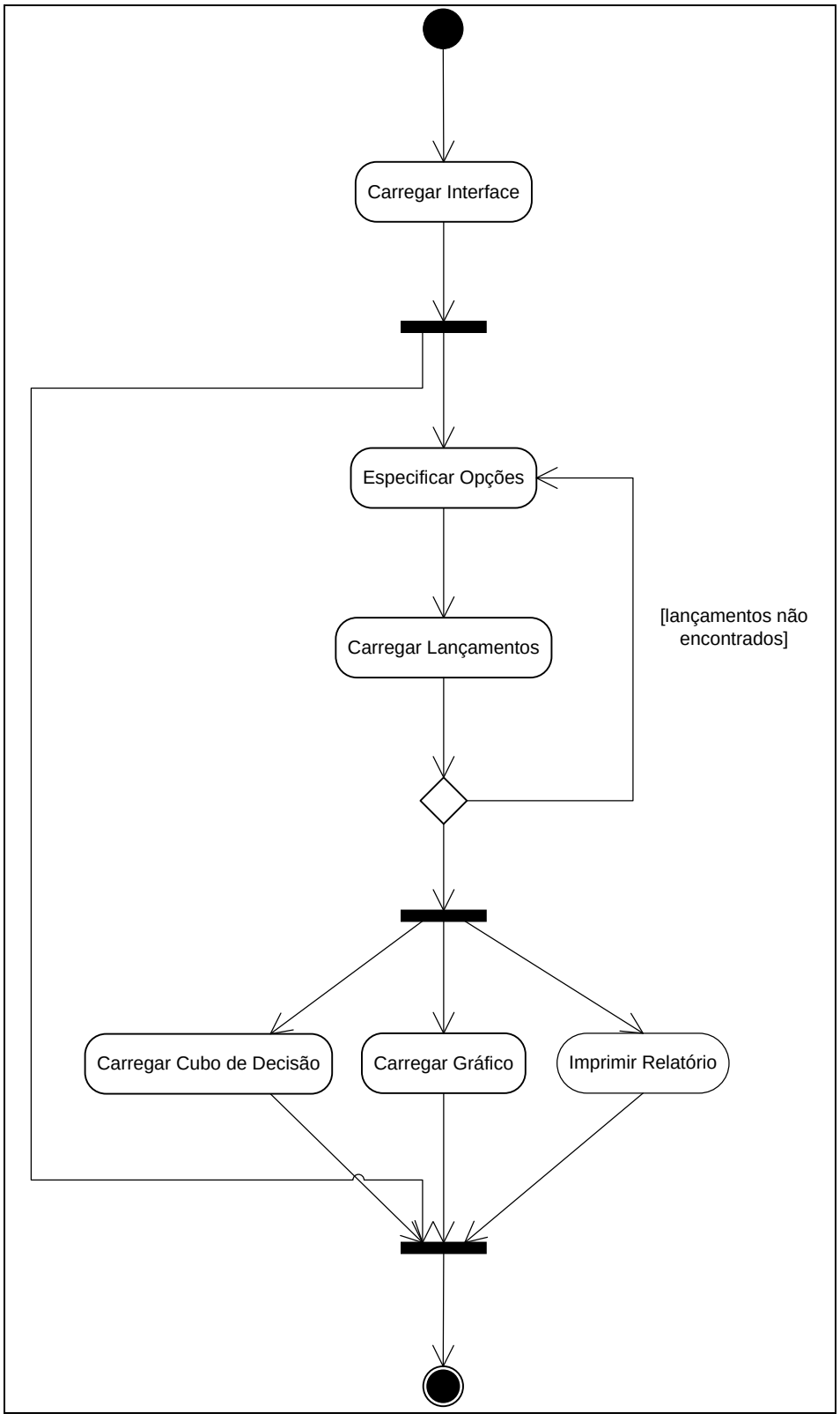


Figura 15 – Diagrama de atividades Consultar Fluxo

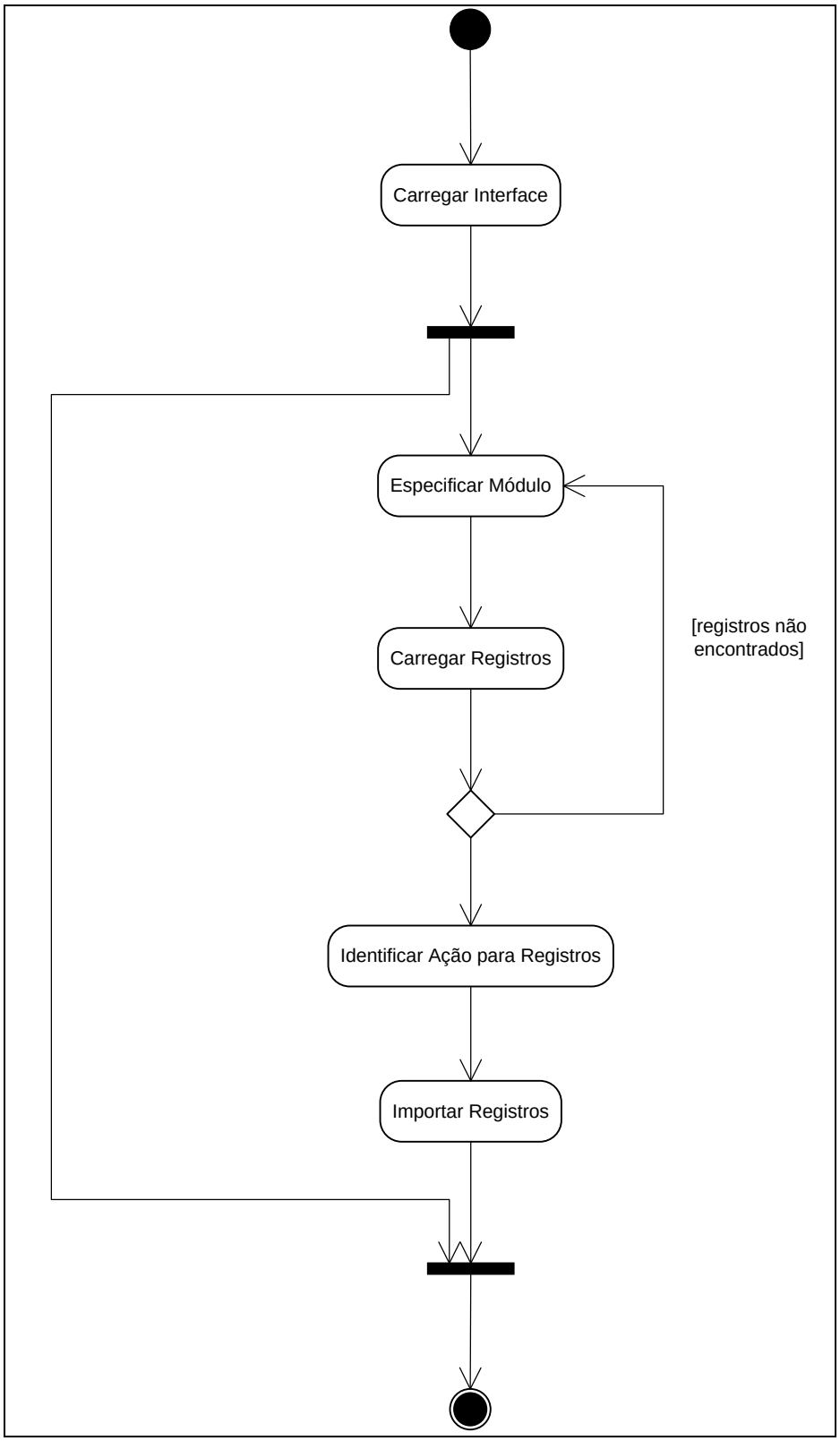


Figura 16 – Diagrama de atividades Importar Dados

b) aplicação servidora

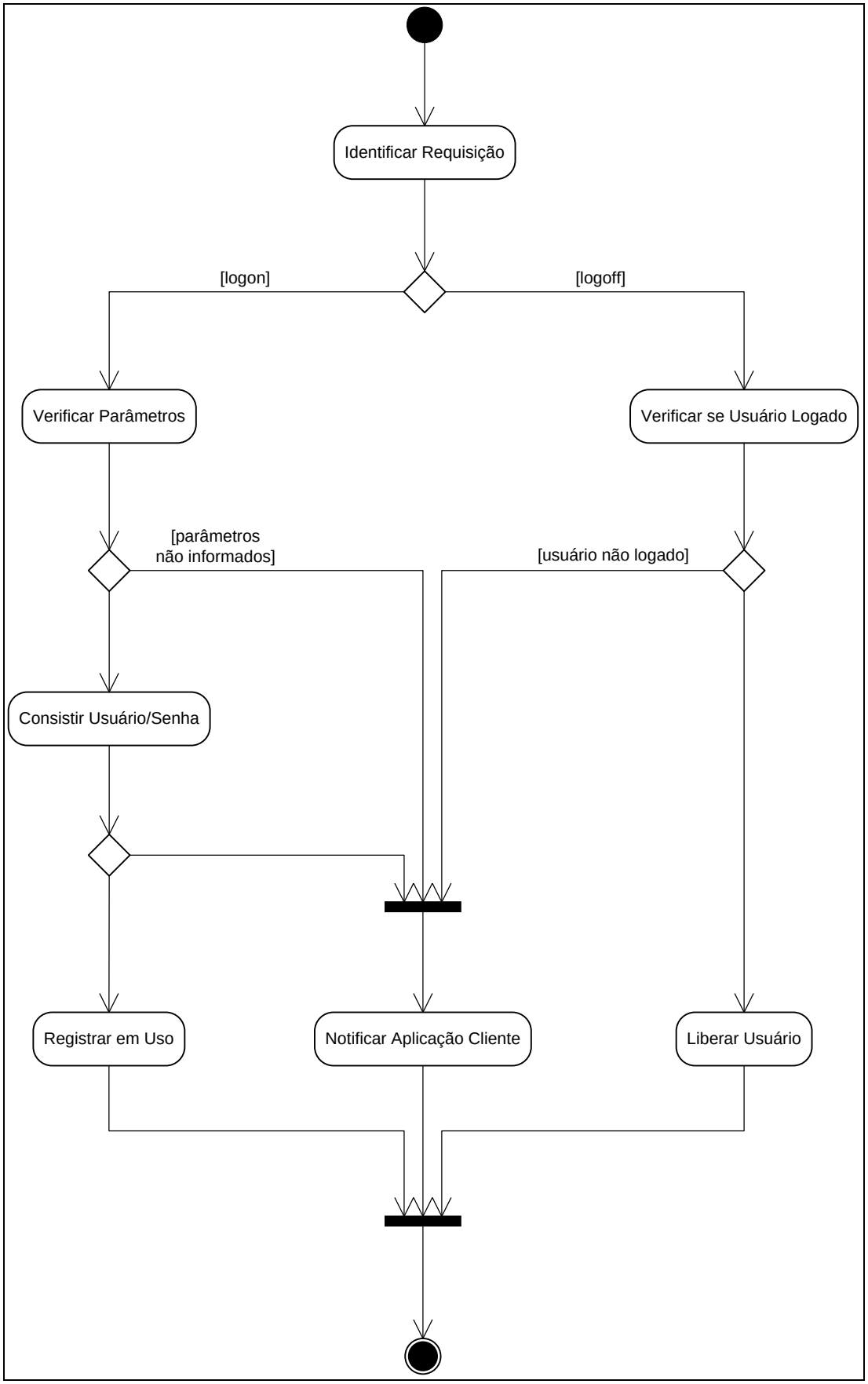


Figura 17 – Diagrama de atividades Autenticar Usuário

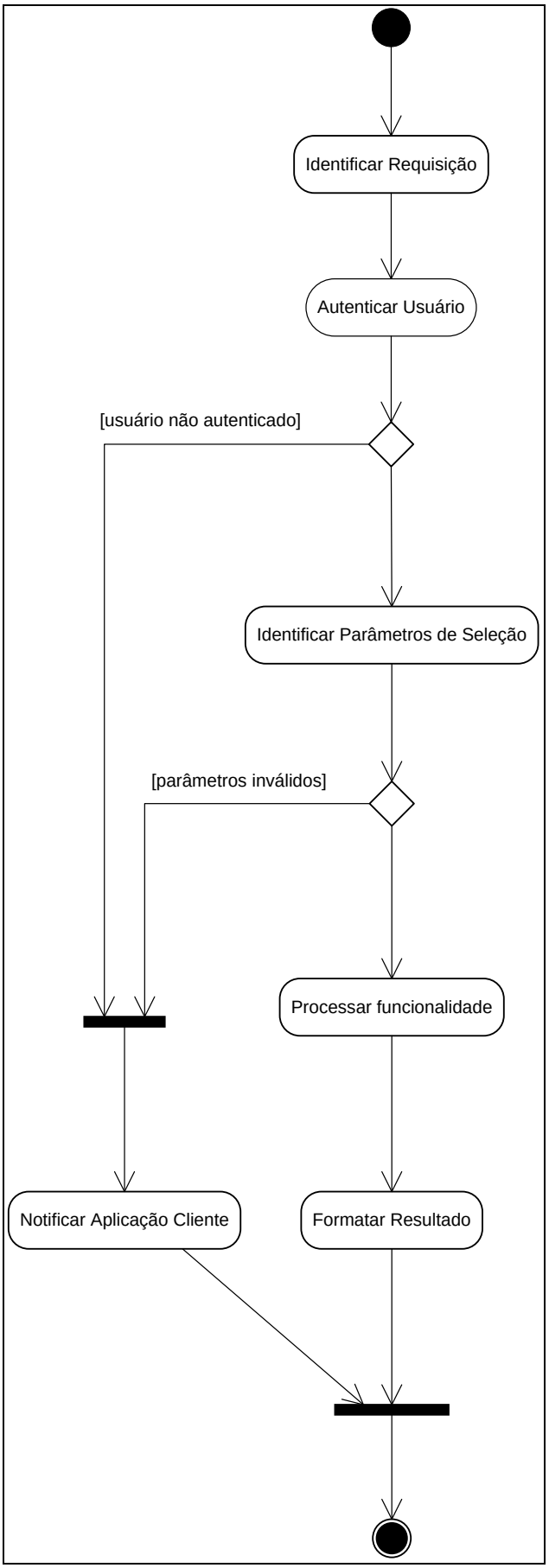


Figura 18 – Diagrama de atividades Gerenciar Lançamentos

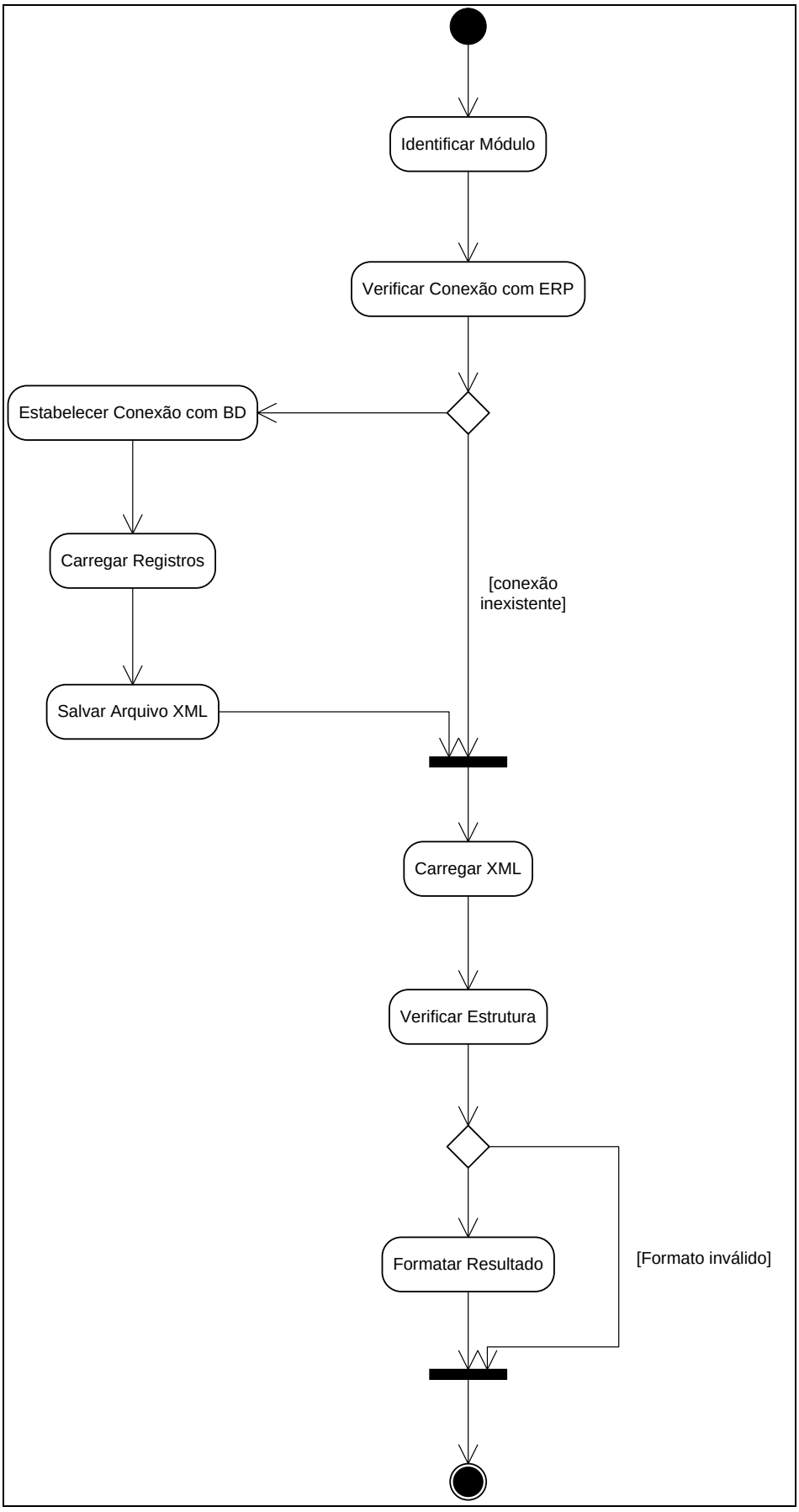


Figura 19 – Diagrama de atividades Carregar Dados do Sistema de Retaguarda

3.4.2 Diagrama entidade-relacionamento

Na Figura 20 está ilustrado o diagrama de entidade-relacionamento da base de dados utilizada para a persistência dos dados utilizados no sistema, especificando as tabelas, seus respectivos campos assim como os relacionamentos entre as entidades.

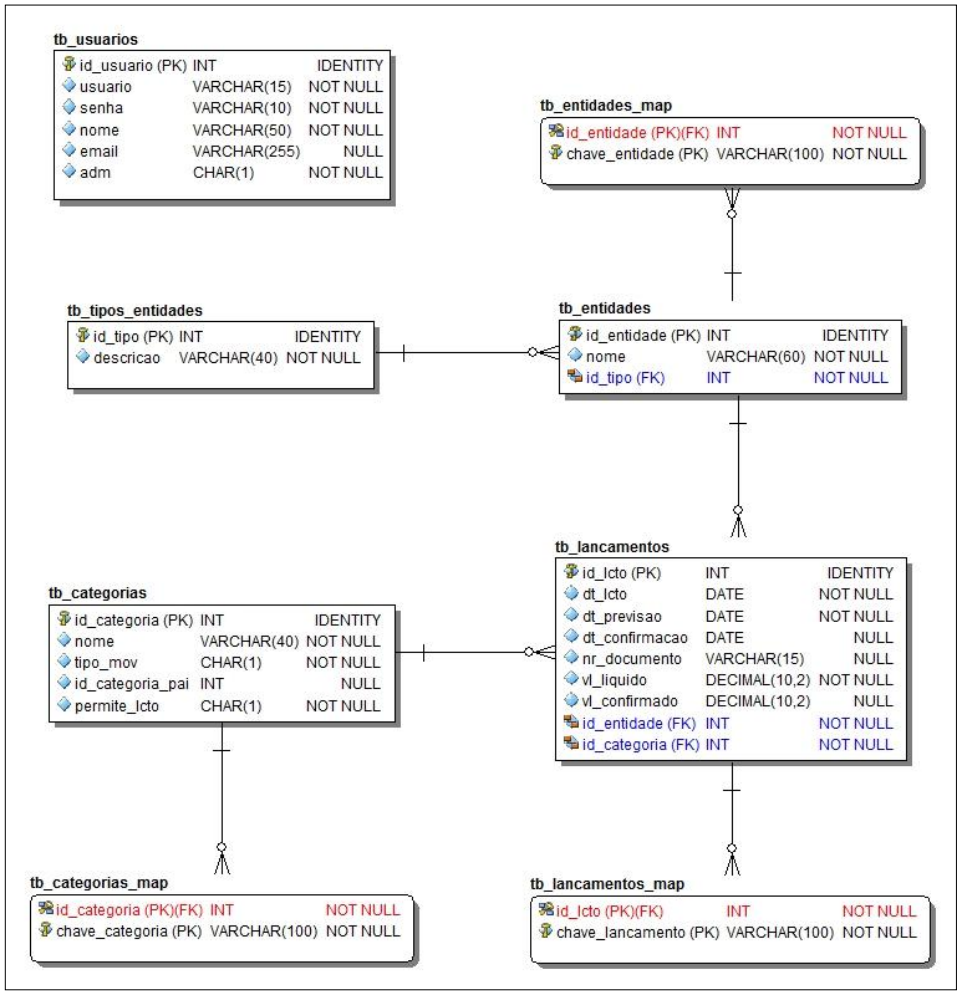


Figura 20 – Diagrama Entidade-Relacionamento

3.4.3 Diagrama de classes

As classes que foram criadas para os serviços *Web* e para a conexão com o sistema de retaguarda estão ilustradas a seguir nas Figuras 21 e 22. As classes de persistência de dados não são ilustradas com diagrama em função de serem

geradas automaticamente pelo *framework* Propel a partir do banco de dados que foi criado, conforme descrito na subseção 4.3.1.

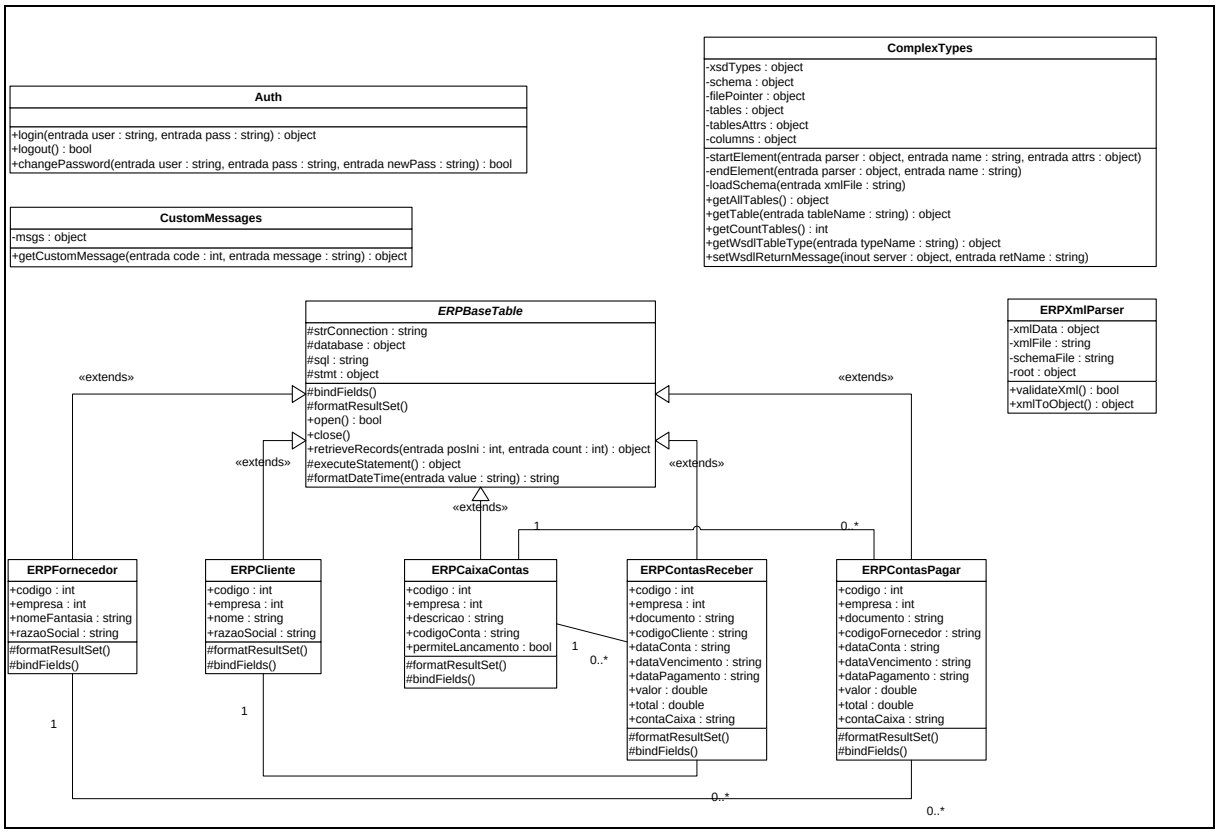


Figura 21 – Diagrama de classes da aplicação servidora

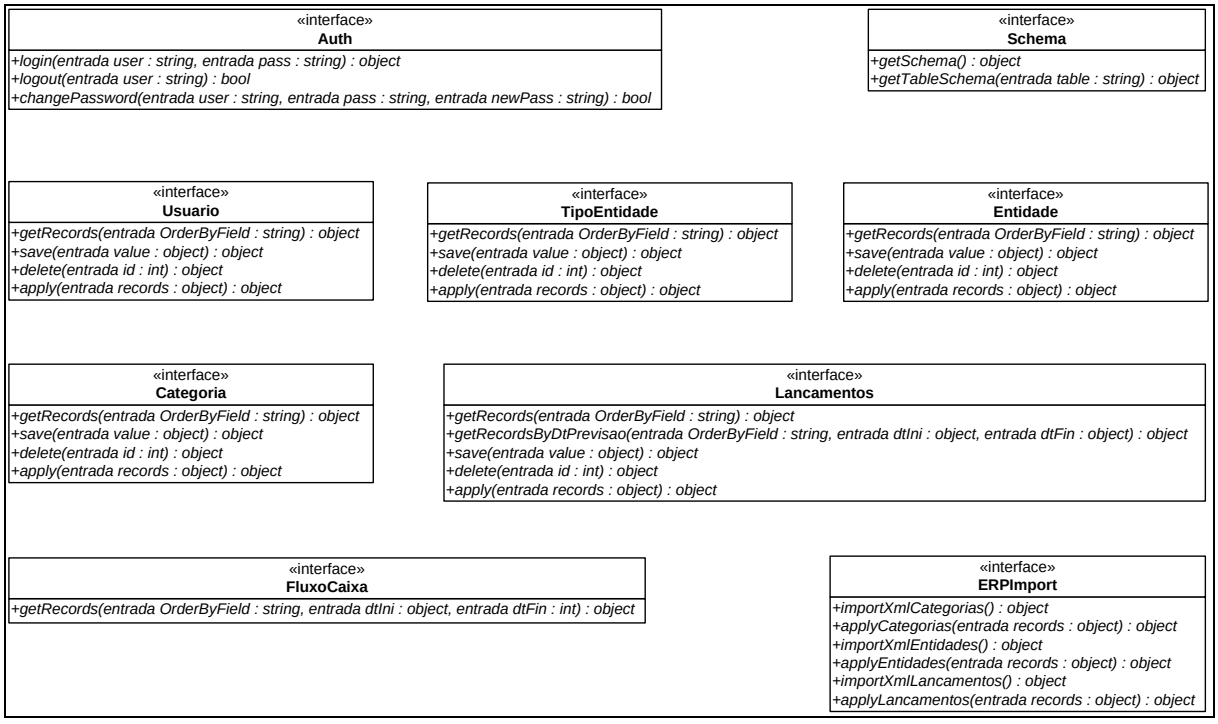


Figura 22 – Diagrama de classes das interfaces dos Web Services

3.5 Testes

Na etapa de testes procura-se identificar possíveis defeitos existentes no aplicativo para que possam ser corrigidos antes que o aplicativo seja implantado no cliente. De acordo com Koscianski e Soares (2007, p. 337):

> O objetivo do teste é encontrar defeitos, revelando que o funcionamento do *software* em uma determinada situação não está de acordo com o esperado. Um teste bem-sucedido identifica defeitos que ainda não foram descobertos e que podem ser, então, corrigidos pelo programador.

O tipo de teste empregado no sistema desenvolvido foi o de caixa-preta, onde são testadas as funcionalidades propostas para o aplicativo e também a *interface* com o usuário. O teste caixa-preta também é conhecido como teste funcional. Ele consiste em testar o comportamento do sistema ou objeto, ignorando a sua construção interna.

> O teste de caixa-preta é baseado nos requisitos funcionais do *software*. [...] O avaliador se concentra nas funções que o software deve desempenhar. A partir da especificação são determinadas as saídas esperadas para certos conjuntos de entrada de dados (KOSCIANSKI e SOARES, 2007, p. 344).

Com o uso do teste de caixa-preta é possível verificar, entre outros problemas:

a) funções incorretas;
b) problemas com a *interface*;
c) desempenho;
d) erros de validação de campos;
e) informações duplicadas na base de dados;
f) impressão de relatórios.

Os testes aplicados envolveram a validação da *interface* de usuário, verificando se as funcionalidades disponíveis estão sendo executadas corretamente e com o resultado esperado. Além disto, foram realizados testes de validação dos campos dos cadastros e movimentações para verificar se os valores informados correspondem ao esperado e também se o aplicativo restringe a digitação de valores incorretos. Juntamente com os testes de *interface* e validação de campos, foram verificados se os incrementos (módulos) atendem aos requisitos levantados e também correspondem às descrições dos casos de uso.

Para organizar a execução dos testes de caixa-preta, fez-se necessário

estabelecer um plano de testes. O plano de testes é fundamental para planejar corretamente a forma com que os testes devem ser conduzidos, a fim de estabelecer um propósito e quais as funcionalidades estarão envolvidas.

Koscianski e Soares (2007, p. 338) entendem que:

> Uma atividade de testes bem organizada pressupõe planejamento. O plano de testes facilita a comunicação entre os envolvidos no desenvolvimento do *software* ao propor um padrão de referência a ser seguido.

O padrão proposto para a realização dos testes de caixa-preta fez uso de casos de teste. Os casos de teste identificam as funcionalidades dentro de cada caso de uso a ser testado, estabelecendo os procedimentos a serem realizados durante os respectivos testes. Desta maneira, os casos de teste servem como um roteiro que deve ser seguido durante o procedimento de teste, identificados por ações e validações que devem ser realizadas.

> Ao tratar programas de computador, o número de possibilidades a serem verificadas pode atingir facilmente uma cifra astronômica. Cada teste da função, contendo um conjunto de entrada diferente – neste caso, composto por um único valor –, é chamado caso de teste.
> Idealmente, os testes devem abranger o maior número de situações possíveis [...]. Escolher bem os exemplos mais representativos de tais situações é uma condição essencial para que os testes contribuam com a verificação de qualidade do programa (KOSCIANSKI e SOARES, 2007, p. 340).

Para a aplicação dos testes, foram seguidos os casos de teste ilustrados nos quadros a seguir:

a) Login

Nome	tcLogin	
Resumo	Testar a funcionalidade de acesso ao sistema	
Ação		**Resultado esperado**
1. Acessar aplicativo		Janela de login
2. Selecionar servidor		
3. Informar “Usuário”		
4. Informar “Senha”		
5. Clicar em “Login”		Carregar Menu do aplicativo. Identificar o usuário logado no menu.
6. Clicar em “Cancelar”		Fechar aplicativo
Validação		**Resultado esperado**
1. Campos obrigatórios (“Servidor”, “Usuário” e “Senha”) não preenchidos		Exibir mensagem de erro. ‘Necessário informar <campo>’
2. Informar ou selecionar “Servidor” inválido		Exibir mensagem de erro. ‘Não foi possível estabelecer conexão com o servidor.’
3. Informar “Usuário” e “Senha” inválidos		Exibir mensagem de erro. ‘Login inválido’

Quadro 15 – Caso de teste login

b) Modelo padrão de cadastros

Nome	tcTemplateCadastro
Resumo	Testas as funcionalidades do template de cadastros
Ação: <u>Inserir registro</u>	**Resultado esperado**
1. Clicar em “Inserir”	Limpar formulário, colocar cadastro em modo de inserção
2. Informar os campos requisitados	
3. Clicar em “Gravar”	Gravar novo registro em memória
4. Clicar em “Cancelar”	Cancela inserção do registro e restaura formulário
Ação: <u>Alterar registro</u>	**Resultado esperado**
1. Localizar registro pelos botões de navegação	Atualizar formulário com os dados do registro
2. Clicar em “Alterar”	Colocar cadastro em modo de alteração
3. Informar ou modificar campos requisitados	
4. Clicar em “Gravar”	Gravar registro modificado em memória
5. Clicar em “Cancelar”	Cancela alteração do registro e restaura formulário
Ação: <u>Excluir registro</u>	**Resultado esperado**
1. Localizar registro pelos botões de navegação	Atualizar formulário com os dados do registro
2. Clicar em “Excluir”	Exibir aviso de confirmação de exclusão. 'Deseja realmente excluir o registro atual?'
3. Clicar em “Sim”	Excluir registro em memória e posicionar no próximo registro do cadastro
4. Clicar em “Não”	Mantém o registro em memória
Ação: <u>Atualizar registros</u>	**Resultado esperado**
1. Clicar em “Atualizar”	Recarregar para a memória os registros da base de dados
Ação: <u>Aplicar modificações</u>	**Resultado esperado**
1. Clicar em “Aplicar”	Gravar modificações realizadas em memória na base de dados Atualizar registros com os identificadores (códigos) dos novos registros
Ação: <u>Desfazer modificações</u>	**Resultado esperado**
1. Clicar em “Desfazer”	Restaurar última ação realizada em memória Restaurar informações do registro
Validação	**Resultado esperado**
1. Gravar registro sem campos obrigatórios preenchidos	Exibir mensagem de erro. ‘Campo <campo> deve ser preenchido’
2. Atualizar dados com modificações pendentes	Exibir mensagem de erro. ‘É necessário aplicar as alterações antes de atualizar os dados’
3. Atualizar dados ou aplicar modificações sem conexão com servidor	Exibir mensagem de erro. ‘Não foi possível estabelecer conexão com o servidor.’
4. Excluir registro referenciado em outro cadastro	Exibir mensagem de erro. ‘Impossível excluir, registro referenciado em outro cadastro’

Quadro 16 – Caso de teste modelo de cadastros

c) Cadastro de usuários

Nome	tcCriarUsuarios	
Resumo	Testar as validações do cadastro de usuários. As ações são as mesmas utilizadas no template de cadastro.	
Validação		**Resultado esperado**
1. Registro duplicado. Não permitir mais de um registro com o mesmo “Username”		Exibir mensagem de erro. ‘Não é possível inserir mais de um registro com o mesmo identificador e/ou chave!’

Quadro 17 – Caso de teste cadastro de usuários

d) Cadastro de tipos de entidade

Nome	tcCadastrarTipoEntidade	
Resumo	Testar as validações do cadastro de tipos de entidades. As ações são as mesmas utilizadas no template de cadastro.	
Validação		**Resultado esperado**
1. Registro duplicado. Não permitir mais de um registro com o mesmo “Nome”		Exibir mensagem de erro. ‘Não é possível inserir mais de um registro com o mesmo identificador e/ou chave!’

Quadro 18 – Caso de teste cadastro de tipos de entidade

e) Cadastro de categorias

Nome	tcCadastrarCategorias	
Resumo	Testar as validações do cadastro de categorias. As ações são as mesmas utilizadas no template de cadastro.	
Validação		**Resultado esperado**
1. Registro duplicado. Não permitir mais de um registro com o mesmo “Nome” e “Tipo de movimentação”		Exibir mensagem de erro. ‘Não é possível inserir mais de um registro com o mesmo identificador e/ou chave!’
2. Categoria não pertence a uma categoria com o mesmo tipo de movimentação		Exibir mensagem de erro. ‘Tipo de movimentação deve ser igual ao "Pertence à".’
3. Atribuir categoria a ela própria		Exibir mensagem de erro. ‘Não pode ser utilizado o mesmo registro como "Pentence à".’

Quadro 19 – Caso de teste cadastro de categorias

f) Cadastro de entidades

Nome	tcCadastrarEntidades
Resumo	Testar as validações do cadastro de entidades. As ações são as mesmas utilizadas no template de cadastro.
Validação	**Resultado esperado**
1. Registro duplicado. Não permitir mais de um registro com o mesmo “Nome” e “Tipo de entidade”	Exibir mensagem de erro. ‘Não é possível inserir mais de um registro com o mesmo identificador e/ou chave!’

Quadro 20 – Caso de teste cadastro de entidades

g) Lançamentos

Nome	tcEfetuarLancamentos
Resumo	Testar as funcionalidades e validações do cadastro de lançamentos. As demais ações são as mesmas utilizadas no template de cadastro.
Ação: <u>Múltiplos lançamentos</u>	**Resultado esperado**
1. Clicar em “Inserir”	Ação “Inserir registro” do template de cadastro. Exibir opções de sequência de lançamentos.
2. Selecionar a frequência desejada	Habilitar quantidade de repetições.
3. Informar a quantidade de repetições	
4. Clicar em “Gravar”	Novos lançamentos criados de acordo com a quantidade de repetições, incrementando a data prevista conforme a frequência selecionada.
Validação	**Resultado esperado**
1. Informar valores negativos em “Valor previsto” e “Valor confirmado”	Exibir mensagem de erro. ‘Conteúdo inválido para o campo.’
2. Informar “Data previsão” inferior a “Data lcto.”	Exibir mensagem de erro. ‘Data não pode ser inferior à do lançamento!’
3. Informar “Data confirmação” inferior a “Data lcto.”	Exibir mensagem de erro. ‘Data não pode ser inferior à do lançamento!’
4. Informar data de lcto., previsão ou confirmação inválida.	Exibir mensagem de erro. ‘Data inválida!’

Quadro 21 – Caso de teste lançamentos

h) Fluxo de caixa

Nome	tcConsultarFluxo
Resumo	Testar a funcionalidades do cadastro de lançamentos. As demais ações são as mesmas utilizadas no template de cadastro.
Ação: <u>Carregar Cubo de Decisão</u>	**Resultado esperado**
1. Selecionar data para o período	
2. Informar data inicial e final	
3. Selecionar a forma de agrupamento	
4. Clicar em “Carregar dados”	Exibir os dados relativos ao período na área do cubo de decisão.
Ação: <u>Imprimir Cubo de Decisão</u>	**Resultado esperado**
1. Executar ação “Carregar Cubo de Decisão”	Habilitar função “Visualizar relatório”.
2. Clicar em “Visualizar relatório”	Abrir janela “Configurar impressão”.
3. Selecionar a impressora	
4. Clicar em “OK”	Exibir a pré-visualização do relatório do cubo de decisão.
Ação: <u>Imprimir Fluxo de Caixa</u>	**Resultado esperado**
1. Selecionar data para o período	
2. Informar data inicial e final	
3. Selecionar a forma de agrupamento	
5. Informar o “Saldo inicial”	
6. Marcar os campos para visualizar	
7. Marcar os valores para visualizar	
8. Clicar em “Visualizar relatório”	Exibir a pré-visualização do relatório do cubo de decisão.
Validação	**Resultado esperado**
1. Informar data inicial ou final inválida.	Exibir mensagem de erro. ‘Data inválida!’
2. Informar data final inferior a data inicial do período.	Exibir mensagem de erro. ‘Período incorreto!’
3. Informar “Data confirmação” inferior a “Data lcto.”	Exibir mensagem de erro. ‘Data não pode ser inferior à do lançamento!’
4. Não informar ao menos um campo e um valor para visualizar.	Exibir mensagem de erro. ‘É necessário selecionar ao menos um campo e um valor para exibir!’
5. Especificar um período que não retorne lançamentos.	Exibir mensagem. ‘Consulta não retornou dados!’

Quadro 22 – Caso de teste fluxo de caixa

i) Importar dados

Nome	tcImportarDados
Resumo	Testar a funcionalidades de importação de dados.
Ação: <u>Carregar registros</u>	**Resultado esperado**
1. Selecionar o que importar: categorias, entidades ou lançamentos.	
2. Clicar em “Carregar registros”	Exibir lista com os registros de importação.
Ação: <u>Marcar todos os registros</u>	**Resultado esperado**
1. Clicar em “Marcar todos”	Selecionar todos os registros na lista.
Ação: <u>Desmarcar todos os registros</u>	**Resultado esperado**
1. Clicar em “Desmarcar todos”	Limpar seleção de todos os registros na lista.
Ação: <u>Aplicar ação aos registros</u>	**Resultado esperado**
1. Selecionar registros na lista ou executar ação anterior “Marcar todos”.	
2. Especificar ação: ignorar ou importar.	
3. Clicar em “Aplicar”	Atribuir a ação específica aos registros selecionados na lista.
Ação: <u>Importar registros</u>	**Resultado esperado**
1. Executar ação “Aplicar ação aos registros”.	
2. Clicar em “Importar registros”.	Gravar registros selecionados no cadastro correspondente. Exibir mensagem “Operação realizada com sucesso!”.
3. Clicar em “OK”.	Limpar a lista de registros visualizados.
Validação	**Resultado esperado**
1. Arquivo de importação (XML) inválido.	Exibir mensagem de erro. ‘Não foi possível realizar a importação! Contate o administrador do sistema para verificar se o arquivo de importação existe e se é válido!’

Quadro 23 – Caso de teste importar dados

j) Alterar senha

Nome	tcAlterarSenha	
Resumo	Testar a funcionalidade de alteração de senha	
Ação		**Resultado esperado**
1. Informar senha atual do usuário autenticado.		
2. Informar “Nova senha”.		
3. Informar “Confirmar senha”.		
4. Clicar em “Confirmar”		Alterar a senha do usuário e voltar para menu.
Validação		**Resultado esperado**
1. Campos obrigatórios (“Usuário”, “Senha”, “Nova senha” e “Confirmar senha”) não preenchidos.		Exibir mensagem de erro. ‘Necessário informar <campo>’
2. Informar “Usuário” e “Senha” inválidos		Exibir mensagem de erro. ‘Login inválido’
3. Informar “Nova senha” e “Confirmar senha” diferentes entre si.		Exibir mensagem de erro. ‘"Nova senha" e "Confirmar senha" devem ser iguais.’

Quadro 24 – Caso de teste alterar senha

3.6 Gerenciamento de tarefas

Por tratar-se de um sistema em pequena escala, não foram utilizadas técnicas específicas para o gerenciamento do projeto. Mas de qualquer forma, houve a necessidade de monitorar a programação do projeto do sistema a fim de acompanhar a situação e distribuição das tarefas.

Sommerville (2003, p. 60) avalia que:

> A necessidade de gerenciamento é uma importante distinção entre o desenvolvimento profissional de *software* e a programação em nível amador. Necessitamos do gerenciamento de projetos de *software* porque a engenharia de *software* profissional está sempre sujeita a restrições de orçamento e de prazo.

No gerenciamento do *software* foi empregada apenas a etapa de programação do projeto que, segundo Sommerville (2003, p.65) “[...] envolve dividir o trabalho total de um projeto em atividades distintas e avaliar o tempo necessário para completar essas atividades”. Sendo assim, para acompanhar e gerenciar as etapas do projeto e suas respectivas tarefas foi utilizado o aplicativo Microsoft

Project, que possibilita a atribuição de tarefas com seus prazos, gerando gráficos de atividades das tarefas envolvidas.

De acordo com Project Homepage (2009, p.1):

> O Microsoft Office Project Standard 2007 fornece ferramentas avançadas de gerenciamento de projeto com a combinação certa de usabilidade, eficiência e flexibilidade, de modo que você possa gerenciar projetos com mais eficiência e eficácia. Você pode se manter informado e controlar o trabalho, as agendas e as finanças do projeto, manter as equipes de projeto alinhadas e ser mais produtivo por meio da integração com programas conhecidos do Microsoft Office system, da geração avançada de relatórios, do planejamento guiado e de ferramentas flexíveis.

As tarefas do projeto foram divididas nas etapas como segue:

a) concepção;
b) elaboração;
c) implementação;
d) testes e correções.

Após agrupadas as tarefas nestas grandes etapas, as mesmas foram subdivididas, conforme apresentado no Diagrama de Gantt e Estrutura Analítica do Projeto.

3.6.1 Diagrama de Gantt

O Diagrama de Gantt trata-se de um diagrama de barras, chamado desta maneira em homenagem ao seu inventor, ele ilustra um calendário do projeto e a data inicial e final das tarefas do projeto (SOMMERVILLE, 2003).

Na Figura 23 pode-se visualizar a estrutura de atividade das tarefas, sua data de início e término, assim como o gráfico de Gantt para o projeto do sistema. As tarefas contidas na etapa de implementação representam os incrementos do processo de desenvolvimento e estão agrupadas em aplicação servidor e aplicação cliente.

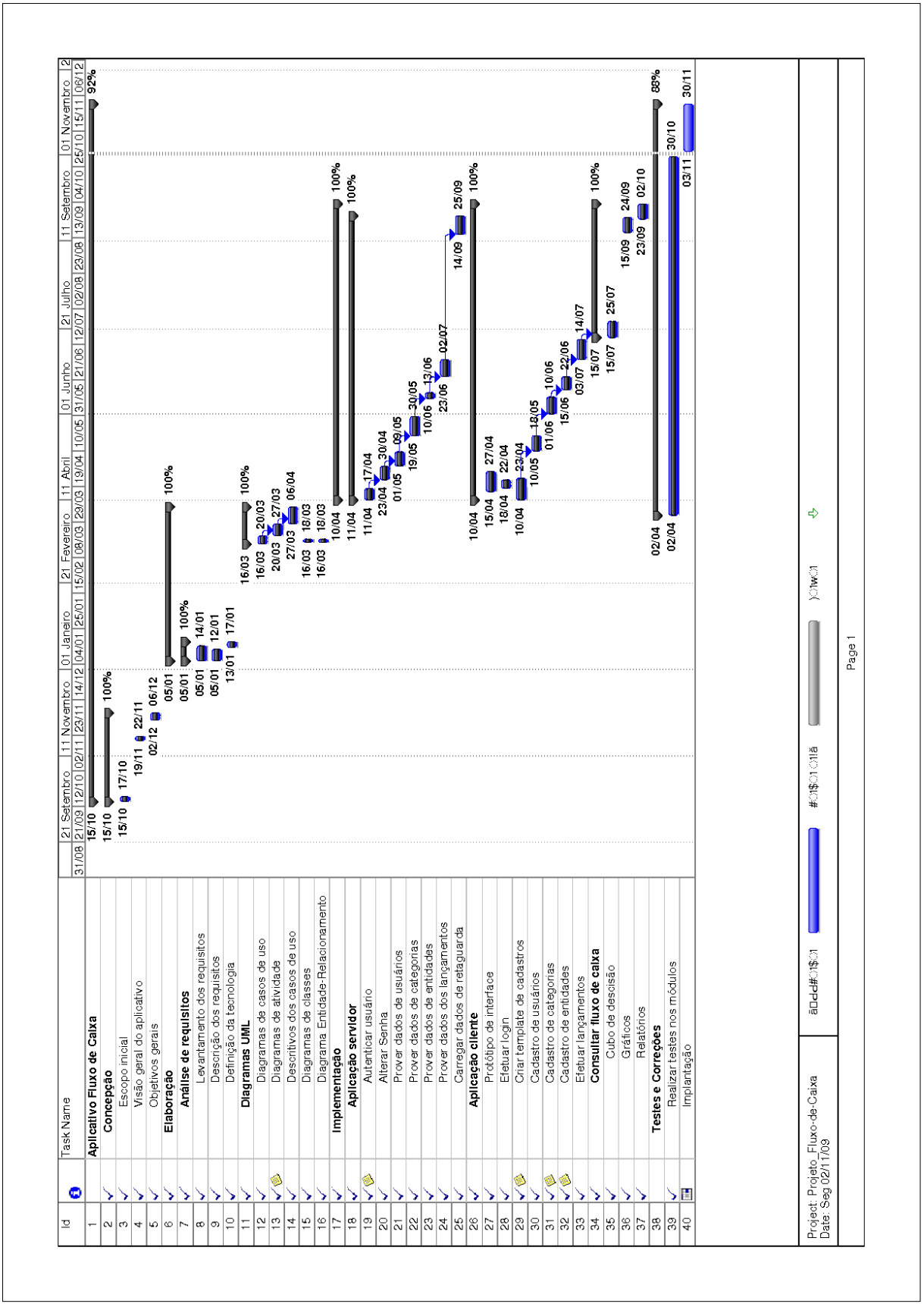


Figura 23 – Diagrama de Gantt do projeto

3.6.2 Estrutura Analítica do Projeto

A Estrutura Analítica do Projeto reflete as tarefas atribuídas ao projeto em uma estrutura de árvore, identificando o percentual concluído das mesmas, assim como outras informações importantes, conforme disponível na Figura 24.

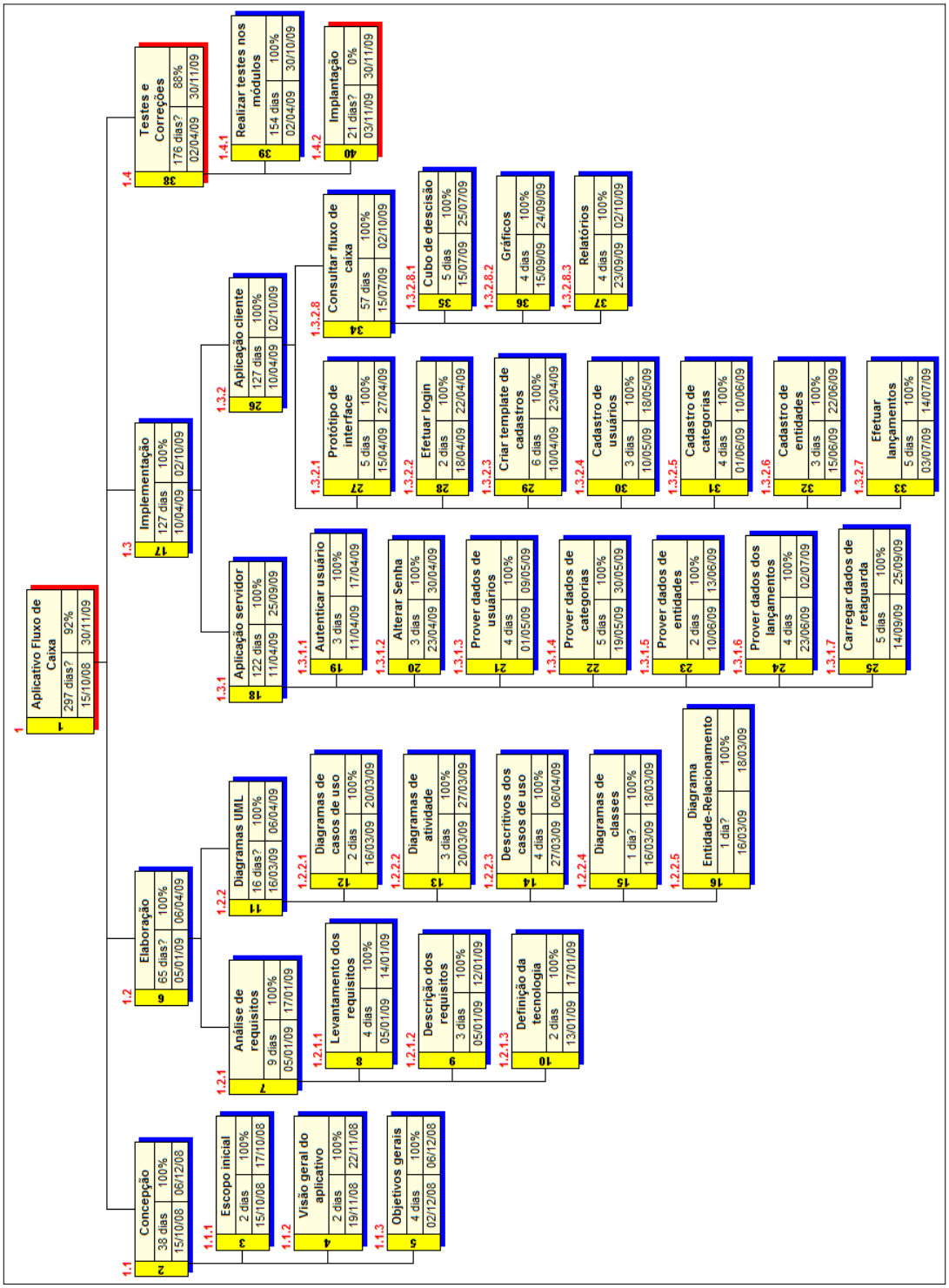


Figura 24 – Estrutura analítica do projeto

4 TECNOLOGIAS

Esta etapa contempla as tecnologias que foram utilizadas na construção e codificação do sistema, assim como as tecnologias que viabilizam o funcionamento do aplicativo, ou seja, são os requisitos necessários para que este seja executado.

O desenvolvimento do aplicativo de fluxo de caixa teve o seu projeto dividido em duas partes. Uma delas foi a construção dos *Web Services*, que são encarregados da conexão com a base de dados do sistema de retaguarda e fornece os serviços necessários para a manipulação e persistência da base de dados requerida pelo aplicativo. A outra parte compreende a aplicação cliente que consume os serviços (*Web Services*) oferecidos e apresenta os resultados ao usuário, ou seja, será a *interface* com o usuário.

No projeto de pesquisa realizado para a construção do sistema, havia sido mencionado que ambas as partes do sistema seriam desenvolvidas utilizando o Delphi e a sua IDE, ambos com plataforma Windows 32 bits, por facilitar a integração entre os Web Services e o aplicativo cliente, ficado transparente a comunicação por intermédio do protocolo SOAP.

No entanto verificou-se a necessidade de criar os *Web Services* em um ambiente que possibilitasse a sua implantação em mais de uma plataforma, tendo em vista que, como o sistema necessita um servidor *Web* para prover os serviços, é complexo limitar a sua instalação somente em um ambiente com servidor Windows, o que é exigido quando se desenvolve *Web Services* em Delphi. Para contornar esta situação, optou-se por desenvolver a aplicação servidora (serviços) utilizando PHP e, neste caso, a sua implantação pode ser realizada em mais de uma plataforma, não restringindo somente a servidores Windows.

A Figura 25 ilustra como funcionam as tecnologias empregadas. Nela podemos visualizar que no aplicativo cliente foi utilizado o Delphi, suas *interfaces* de serviço para conexão com os serviços *Web*, assim como os componentes para geração de relatórios e cubo de decisão. Do lado do servidor, o servidor HTTP Apache, o interpretador PHP, com a extensão NuSOAP e o *framework* Propel e, também, o SGBD MySQL.

Estas tecnologias são descritas nas subseções seguintes.

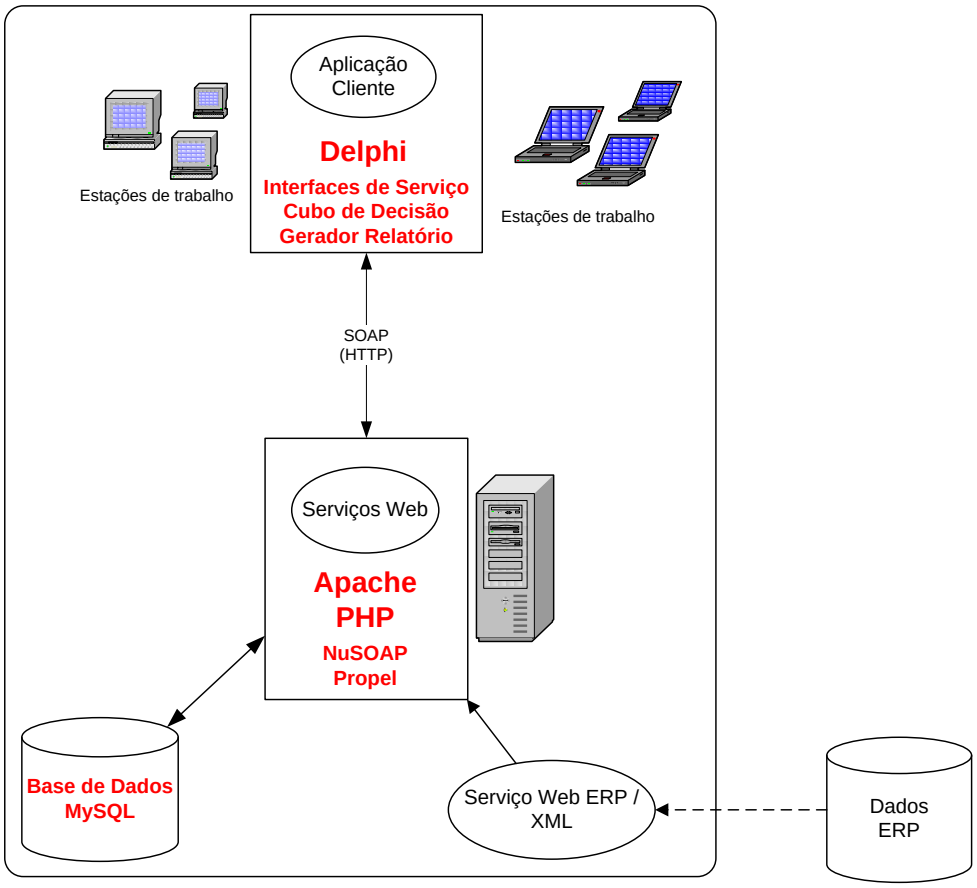


Figura 25 – Tecnologias utilizadas

4.1 Servidor HTTP Apache

O servidor HTTP Apache é mais conhecido por servidor Apache, em função da Apache Software Foundation, que é a fundação responsável por manter e desenvolver diversos projetos de *software* livre, dentre eles o servidor *Web* HTTP. Foi criado em fevereiro de 1995 por Rob McCool na NCSA (*National Center for Supercomputing Applications*), Universidade de Illinois e a sua versão 1.0 foi lançada em dezembro de 1995. O servidor Apache tem sido o mais popular servidor web na *Internet* desde abril de 1996, principalmente em função de ser um *software* livre e poder ser instalado em diversos sistemas operacionais. (HTTP SERVER PROJECT, 2009).

O servidor Apache tem suas funcionalidades estruturadas em módulos que podem ser integrados ao servidor, como módulo específico para autenticação e

suporte a linguagens de programação, por exemplo, e podem ser criados por desenvolvedores independentes. Um dos módulos mais comuns, que normalmente é integrado ao Apache, é o interpretador de scripts do PHP, conforme descrito em subseção específica.

A versão utilizada nos testes realizados para a implementação do sistema foi a versão 2.2.11 de dezembro de 2008.

4.2 SGBD MySQL

O MySQL é um sistema gerenciador de banco de dados relacional *open source* (código fonte aberto) que se tornou o mais popular em seu segmento devido a sua performance, alta confiabilidade e facilidade de uso. Ele utiliza a linguagem SQL (*Structured Query Language* – linguagem de consulta estruturada) para manipulação e consulta dos dados e é amplamente utilizado em aplicações *Web*, podendo ser instalado em mais de 20 plataformas diferentes. (MYSQL, 2009).

Dentre as principais características estão:

a) portabilidade, a capacidade de ser instalado em plataformas diferentes;
b) compatibilidade com diversas linguagens de programação, através de *drivers* (*software* de comunicação) específicos para interligação com o SGBD;
c) licença gratuita para uso;
d) restrição de acesso por intermédio de privilégio de usuários;
e) suporte a mais de uma conexão simultânea, permitindo que mais de um usuário esteja conectado ao mesmo tempo, gerenciando as transações que são submetidas.

O sistema desenvolvido fez uso da versão 5.1.34 do MySQL, de abril de 2009. Sua escolha ocorreu em função de sua ampla utilização nos servidores web e de sua fácil integração com a linguagem PHP e o *framework* Propel, descritos nas subseções seguintes.

4.3 PHP: Hypertext Preprocessor

Conforme o Manual do PHP (2009):

> PHP, que significa "PHP: Hypertext Preprocessor", é uma linguagem de programação de ampla utilização, interpretada, que é especialmente interessante para desenvolvimento para a Web e pode ser mesclada dentro do código HTML. A sintaxe da linguagem lembra C, Java e Perl, e é fácil de aprender. O objetivo principal da linguagem é permitir a desenvolvedores escreverem páginas que serão geradas dinamicamente rapidamente, mas você pode fazer muito mais do que isso com PHP.

Em outras palavras, PHP é uma linguagem interpretada e amplamente utilizada no desenvolvimento de páginas *Web* dinâmicas, pois permite que seu código seja carregado juntamente com o código HTML.

A linguagem teve sua origem em 1994, sendo criada por Rasmus Lerdorf que criou um conjunto de scripts voltados para a criação de páginas dinâmicas usada para controlar o acesso ao seu currículo através da *Internet*. (DALL'OGLIO, 2007).

De acordo com Minetto (2007, p. 14):

> Uma das grandes vantagens do PHP é sua facilidade de aprendizado. Ao ler poucas páginas de tutoriais ou de algum livro, um programador já é capaz de montar um formulário HTML e de criar um script PHP que processe os dados fornecidos pelo usuário.

Isso é facilmente constatado, pois a estrutura do código PHP é muito fácil de ser compreendido, sem a necessidade de uma grande experiência em programação, favorecendo a sua grande utilização em aplicações voltadas para *Web*.

PHP inicialmente foi criado para ser uma linguagem de script estruturada, mas devido sua ampla utilização, novos recursos e funcionalidades foram sendo incorporados à linguagem, tendo como principal objetivo torná-la uma linguagem orientada a objetos (MINETTO, 2007).

Por ser uma linguagem interpretada, o código fonte necessita ser processado pelo servidor HTTP, que é quem irá receber as requisições dos usuários, por intermédio do *browser* (navegador *Web*), ao acessar determinado *script* e então processar o código fonte do *script* PHP e depois devolvê-lo ao *browser* sob forma de HTML e este apresenta ao usuário que o havia requisitado.

Dentre os principais motivos pela escolha do PHP está a sua extensibilidade, conexão com vários bancos de dados, suporte à orientação a objetos e sua ampla presença nos domínios da *Internet* e em *Intranet* empresariais.

4.3.1 Framework Propel

Propel é um *framework* ORM (*Object-Relational Mapping* – mapeamento objeto-relacional) para PHP, versão 5. Ele possibilita o acesso a bases de dados usando um conjunto de objetos, proporcionando uma API (*Application Programming Interface* – interface de programação de aplicativos) para gravar e obter dados. (PROPEL, 2009).

Minetto (2007, p. 17) descreve *framework* da seguinte maneira:

> [...] é uma "base" de onde se pode desenvolver algo maior ou mais específico. É uma coleção de códigos-fonte, classes, funções, técnicas e metodologias que facilitam o desenvolvimento de novos *softwares*.

A utilização de um *framework* para desenvolvimento oferece algumas vantagens, dentre elas estão o uso de convenções, classes e bibliotecas, assim uma equipe de desenvolvedores segue um mesmo padrão, facilitando a manutenção do sistema. Outra vantagem está na automatização de tarefas repetitivas, como operações de inserção, exclusão e modificação de dados em tabelas, que normalmente ocorrem de forma similar, variando apenas nos campos envolvidos.

A necessidade de persistir os dados é ponto fundamental em um sistema, Dall'Oglio (2007, p. 221) entende que "quando trabalhamos com um conjunto de objetos, precisamos persistir estes objetos da base de dados, ou seja, armazená-los e permitir posterior recuperação.".

Dall'Oglio (2007, p. 221) complementa sua definição sobre persistência de dados da seguinte maneira:

> [...] a persistência significa a possibilidade de esses objetos existirem em um meio externo à aplicação que os criou, de modo que esse meio deve permitir que o objeto perdure, ou seja, não deve ser um meio volátil. Os bancos de dados relacionais são o meio mais utilizado para isso (embora não seja o único).

Neste sentido é que se faz necessário o uso do mapeamento objeto-relacional, ou seja, realizar a tarefa de traduzir (mapear) objetos de negócio em tabelas do banco de dados e vice-versa. Para realizar esta tarefa foi utilizado o *framework* Propel realizando o mapeamento de forma automatizada, sendo assim, não há necessidade de manipulação dos dados utilizando linguagem SQL, isto é realizado pelo *framework* que são classes da aplicação que consistem uma camada específica de acesso aos dados.

De acordo com Propel (2009) o *framework* Propel possibilita aos desenvolvedores web trabalhar com bases de dados da mesma maneira que trabalham com classes e objetos em PHP, dentre suas principais características, destacam-se:

a) não é preciso se preocupar com a conexão com base de dados ou instruções SQL;
b) não há necessidade de realizar conversão de dados ou tipos de dados;
c) a definição da base de dados é feita em um arquivo XML e o Propel irá criar a base de dados e gerar as classes e objetos que suportem a *interface* com a base de dados gerada.

Para o seu funcionamento, o Propel tem alguns requisitos, sendo eles:

a) PHP versão 5;
b) um SGBD relacional suportado pelo *framework*. Atualmente o Propel apresenta suporte à MySQL, PostgreSQL, SQLite e MS SQL Server;
c) PEAR (*PHP Extension and Application Repository*) – um *framework* e sistema utilizado para distribuir componentes reutilizáveis para PHP;
d) PEAR::Log – um pacote que proporciona o *log* do *framework*, incluindo saídas dos manipuladores para arquivos, registro de *log* da base de dados, *e-mail*, etc;
e) Creole – Propel utiliza o Creole como a camada de abstração da base de dados. As classes Creole são incluídas e distribuídas juntamente com os pacotes Propel;
f) Phing – usado pelo *framework* para construir as classes PHP e arquivos de definição SQL. Phing é utilizado somente no ambiente de desenvolvimento do Propel, mas não é requerido para a distribuição do sistema.

A Figura 26 ilustra, com um diagrama, a arquitetura utilizada pelo Propel. Nela pode-se visualizar a integração dos *scripts* PHP com o Propel e este comunicando com a API Creole, que se encarrega de interagir à base de dados configurada.

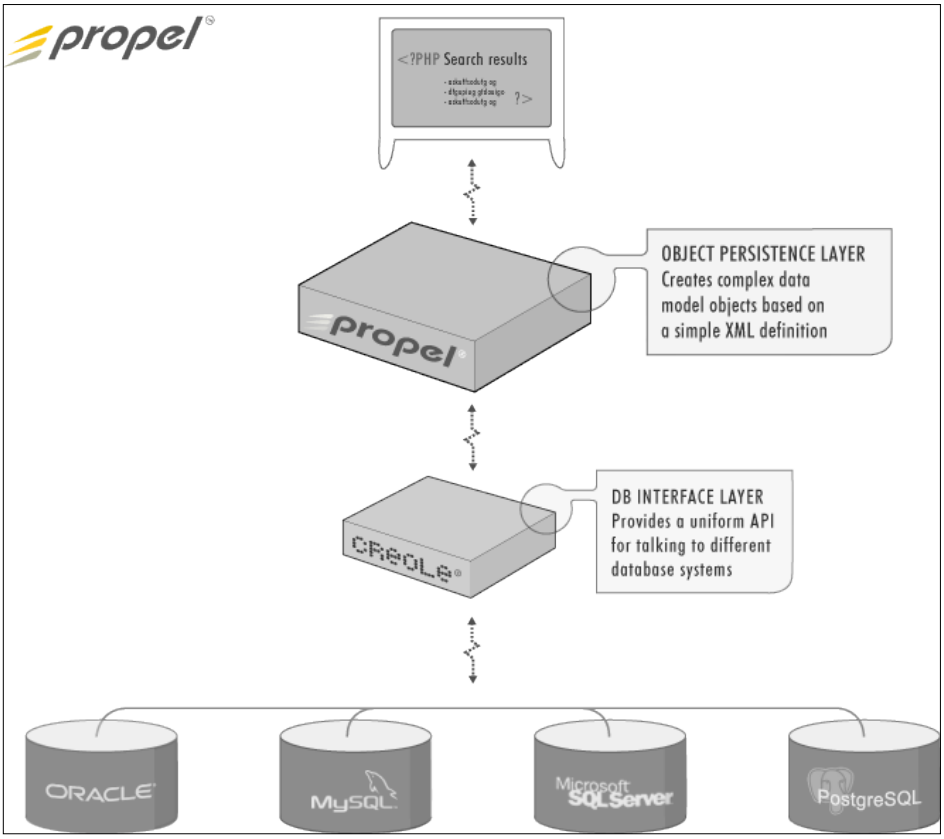


Figura 26 – Diagrama da arquitetura do framework Propel
Fonte: Propel (2009, p. 1)

Para o desenvolvimento do sistema foi utilizado o gerador do Propel para realizar a engenharia reversa da base de dados que foi criada em um primeiro momento. Depois de realizada a engenharia reversa, o Propel cria um arquivo XML contendo o esquema da base de dados. Este arquivo pode ser modificado para atender certas necessidades específicas, como nome de tabelas ou campos, e após efetuados os ajustes, quando necessário, é utilizado novamente o gerador do Propel para criar a estrutura de classes da aplicação encarregadas de realizar o mapeamento e persistência dos dados.

4.3.2 NuSOAP

NuSOAP é um conjunto de classes para PHP que permite ao desenvolvedor criar e consumir serviços *Web* baseados em SOAP 1.1, WSDL 1.1 e HTTP 1.0/1.1, sem a necessidade de qualquer extensão PHP (NUSOAP, 2009).

Para o NuSOAP funcionar, não há necessidade de instalação de nenhum

pacote, pois, como ele é um conjunto de classes, basta que seus arquivos fontes sejam distribuídos e associados aos demais *scripts* PHP, ou seja, para usar a biblioteca NuSOAP, é necessário incorporar ao código da aplicação a vinculação do módulo NuSOAP.

A opção por utilizar o NuSOAP para a implementação dos serviços *Web* se deve em função da facilidade de criar e registrar os serviços, assim como publicar as funcionalidades e tipos complexos de dados através dos documentos WSDL. Com o uso de NuSOAP, não há necessidade de criar documentos WSDL como arquivos específicos, pois ao implementar o servidor SOAP basta executar métodos específicos para gerar e publicar os tipos de dados e registro dos serviços contidos no WSDL.

Para o aplicativo servidor foi criado um serviço *Web*, utilizando o NuSOAP, para cada caso de uso do sistema, permitindo que possa ser carregado de forma isolada, sem a necessidade de carregar todas as *interfaces* dos serviços de uma só vez.

4.4 Delphi

Delphi é uma linguagem de programação baseada em *object pascal*, o que possibilita a programação orientada a objetos, recurso disponível nas linguagens de programação mais modernas. O *object pascal* é uma extensão orientada a objetos da linguagem Pascal que é essencialmente estruturada (CANTU, 2003).

O Borland Delphi 7 Developer's Guide (2002) descreve o Borland Delphi como um ambiente de programação visual orientado a objeto, para desenvolvimento de aplicações 32-bit. Com Delphi é possível criar aplicações altamente eficientes com um mínimo de codificação manual.

O *object pascal*, que é a base da linguagem Delphi, é uma linguagem orientada a objetos, não pura, mas híbrida por possuir características de programação estruturada e por suportar os tipos primitivos de dados como, *integer, string,* por exemplo, sem a necessidade de instanciar um objeto para o uso destes tipos de dados.

Podem-se destacar algumas características da linguagem, como a fácil leitura

do código fonte, que é mantido organizado e de fácil entendimento, o compilador rápido, permite o uso do código fonte de forma modular e a possibilidade de gerar aplicativos executáveis sem a necessidade de distribuição de qualquer outro tipo de extensão para que o programa seja executado.

Segundo Cantu (2003, p. 3) “Em uma ferramenta de programação visual como o Delphi, o papel do IDE [...] é, às vezes, ainda mais importante do que a linguagem de programação”, e sob esse aspecto, um dos pontos fortes do Delphi é a sua IDE (*Integrated Development Environment* – ambiente integrado de desenvolvimento) que possibilita um local único para o desenvolvimento de uma forma geral. Através da IDE do Delphi temos acesso ao código fonte do projeto e a todos os arquivos que o compõe. Por intermédio de componentes disponibilizados na sua VCL (*Visual Component Library* – biblioteca de componentes visuais) e da técnica RAD, é possível programar a *interface* gráfica do aplicativo de forma visual, tornando o desenvolvimento ágil, principalmente para o protótipo da aplicação.

Quando o Delphi é iniciado, o desenvolvedor é levado imediatamente para a IDE, que provê acesso a todas as ferramentas necessárias para projetar, desenvolver, testar, depurar e distribuir aplicações, permitindo uma rápida prototipagem em um curto espaço de tempo. (BORLAND DELPHI 7 DEVELOPER’S GUIDE, 2002).

A IDE do Delphi permite que componentes de terceiros ou criados pelo próprio desenvolvedor sejam adicionados, enriquecendo ainda mais a biblioteca de componentes e agilizando o processo de desenvolvimento de aplicativos, pois, com a utilização de componentes prontos e testados, há a redução de tempo de desenvolvimento e testes.

Um dos recursos mais utilizados em programação, de uma forma geral, é a depuração do código fonte enquanto o aplicativo está em execução. Neste aspecto, a depuração de código do projeto no ambiente do Delphi é um dos recursos mais bem explorados pela ferramenta, pois possibilita a inspeção de variáveis, o estado de determinados objetos e acompanhar passo a passo a execução do aplicativo, tanto local quanto remotamente.

O ambiente do Delphi permite o desenvolvimento de aplicativos tanto para a plataforma Win32 (Microsoft © Windows 32btis), quanto para a plataforma .Net da Microsoft, por intermédio do *.Net Framework*, disponibilizado pela Microsoft e integrado a IDE do Delphi.

4.4.1 Interfaces de serviço

De acordo com Borland Delphi 7 Developer's Guide (2002), todo servidor que implemente serviços *Web* são construídos utilizando *invokable interfaces* (tratado aqui como *interfaces* de serviço), que são as *interfaces* utilizadas para realizar as chamadas (invocar) o serviço *Web*. As *invokable interfaces* contêm as informações a respeito dos tipos de dados dos objetos, que são os tipos complexos contidos no WSDL, e nos clientes (consumidores). As informações dos objetos enquanto estão em execução são usadas para gerar dinamicamente uma tabela com os métodos disponíveis e para realizar as chamadas a estes métodos que estão contidos na *interface*.

Usar o Delphi para desenvolver uma aplicação cliente que acesse um serviço *Web* é um processo relativamente simples, pois existem ferramentas e funcionalidades específicas para estabelecer a conexão com o servidor por intermédio do protocolo SOAP.

Cantù (2003, p. 710) descreve essa facilidade da seguinte maneira:

> Especificamente, o Delphi fornece um mapeamento bidirecional entre WSDL e *interfaces*. Isso significa que você pode pegar um arquivo WSDL e gerar uma *interface* para ele. Em seguida, você pode criar um programa-cliente incorporando pedidos SOAP por meio dessas *interfaces* e usar um componente Delphi especial que permite converter as requisições de sua *interface* local em chamadas SOAP [...].

O que Cantù afirma é que o Delphi oferece um mecanismo chamado *WSDL Import Wizard* (assistente de importação de WSDL), para carregar o conteúdo de um documento WSDL e gerar e registrar automaticamente uma *interface* para o serviço *Web*, permitindo consumir estes serviços fornecidos através das chamadas aos métodos publicados.

Na Figura 27 está ilustrado o assistente em execução, realizando a importação de um WSDL do sistema. Após realizada a importação, o Delphi cria um arquivo fonte com o mesmo nome do publicado no serviço *Web* contendo a declaração da *interface* de serviço, os tipos de dados publicados no WSDL. Também é criada uma função global específica para realizar a chamada do serviço retornando a *interface* apropriada (*invokable interface*), assim como o registro da *interface* de serviço para utilização no aplicativo cliente.

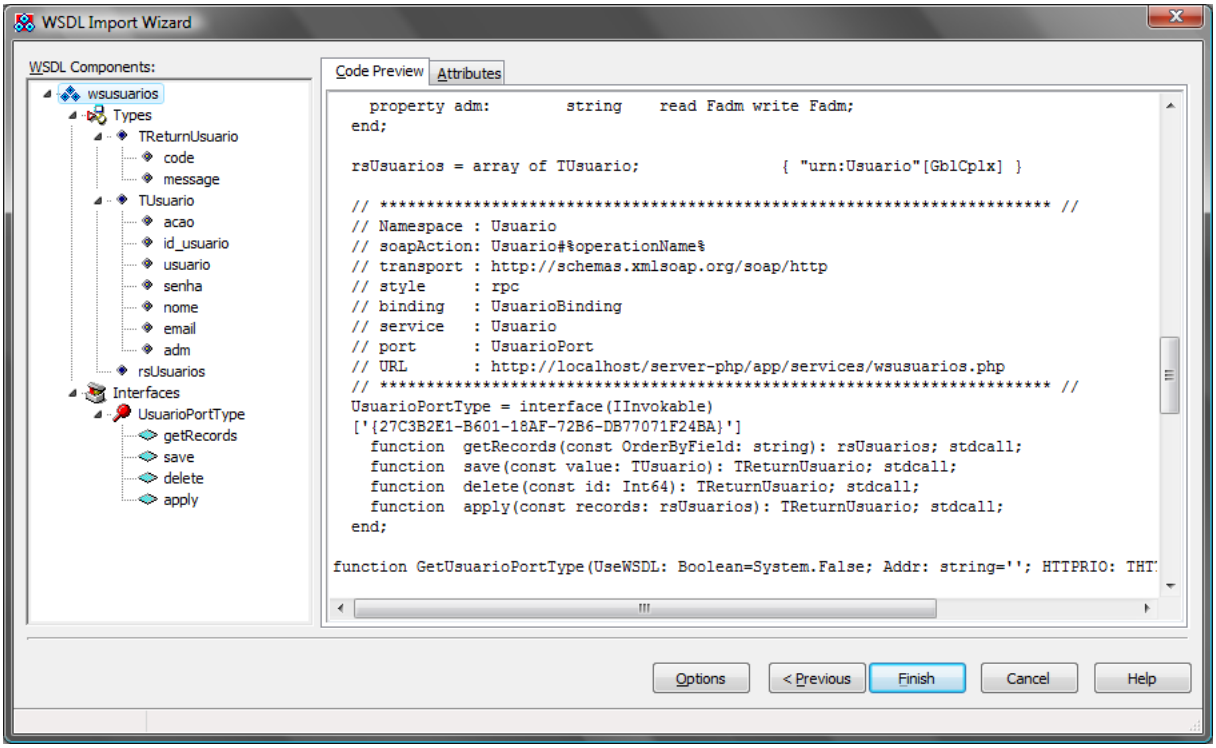


Figura 27 – Assistente de importação de WSDL

O componente mencionado por Cantù para realizar a conversão das requisições locais em chamadas SOAP, é o HTTPRIO da classe THTTPRIO que processa as chamadas. Ele realiza a conexão com o serviço fornecendo o URL (*Uniform Resource Locator* – localizador de recurso universal, o endereço do recurso) do documento WSDL para extrair o endereço da chamada SOAP ou então fornecer diretamente o URL da chamada SOAP.

Cada um dos serviços *Web* que foram publicados utilizando o PHP no servidor, por intermédio do NuSOAP, foram carregados na aplicação cliente, gerando assim uma *interface* de serviço isolada e invocada pelo aplicativo cliente à medida em que o usuário necessita da informação de determinada parte do sistema, o caso de uso em questão.

Uma vez que o usuário invoque (requisite) as informações a respeito de determinado serviço, ou seja, os registros relacionados a determinado cadastro ou conjunto de lançamentos, ele estará realizando uma requisição ao servidor e carregando para a memória do aplicativo cliente os dados na forma em que foram publicados e registrados na *interface*. Uma vez carregados para a memória, esses dados são manipulados, realizando operações de inserção, exclusão ou modificação, e então são devolvidos ao servidor, por intermédio de novas chamadas às *interfaces*, enviando o conjunto de dados manipulado sob forma de parâmetro,

também respeitando os tipos de dados registrados.

4.4.2 PivotCube

PivotCube é um conjunto de componentes para Delphi que permite a geração de cubos de decisão OLAP (*On-Line Analytical Processing* – processamento analítico on-line) a partir de um conjunto de dados disponibilizado.

PivotCube VCL (2009) define a tecnologia PivotCube como uma das melhores implementações OLAP para análise de dados multidimensional, combinando a flexibilidade de uma API que o programador precisa, com a conveniência que permite ao usuário final realizar de uma forma eficiente suas tarefas de análise de dados multidimensional, usando uma análise estatística dos dados correntes de uma base de dados relacional. Além disso, possibilita que o usuário final faça, enquanto estiver executando o aplicativo, seus próprios totalizadores, agrupamento de dados (linhas e colunas), entre outras funcionalidades.

Os componentes PivotCube fornecem uma *interface* gráfica amigável ao usuário, facilitando as modificações necessárias no cubo de decisão para que atendam as suas necessidades, quanto ao conjunto de dados, subtotais, gráficos, etc.

PivotCube VCL (2009) apresenta algumas funcionalidades disponíveis, algumas delas:

a) mover colunas nos gráficos;
b) totais por linhas e/ou colunas;
c) dimensões multinível em forma de árvore;
d) funções estatísticas para média, desvio e variância;
e) gráficos embutidos;
f) inclusão ou exclusão das dimensões do cubo;
g) filtragem dos dados por dimensões ou valores;
h) ordenar os dados apresentados;
i) exportar de dados para Excel.

O PivotCube requer a distribuição de uma biblioteca (arquivo DLL) juntamente com o aplicativo cliente para que este carregue os dados para o cubo. O

componente é gratuito e pode ser distribuído livremente. Seu uso fica limitado até um conjunto de 5000 registros carregados para o cubo, acima desta quantidade o PivotCube passa a operar em modo *Trial*[2] exibindo uma tela de aviso.

4.4.3 FastReport

FastReport é um pacote de componentes para a VCL do Delphi que possibilita a geração, visualização e impressão de relatórios.

O FastReport é considerado um conjunto de componentes que permite que a aplicação gere relatórios de forma rápida e eficiente. Ele disponibiliza todas as ferramentas necessárias para desenvolver relatórios, incluindo um ambiente visual de formatação do relatório e uma janela de pré-visualização. (FASTREPORT VCL, 2009).

O FastReport possui uma série de funcionalidades bastante úteis para a apresentação de relatórios, dentre elas:

a) relatórios multipáginas – impressão de relatórios com mais de um formato de página;
b) sub-relatórios – possibilidade de dividir o relatório em unidades menores, embutindo-as em páginas do relatório;
c) agrupamento de registros, permitindo a criação de relatórios multiníveis – mestre/detalhe, mestre/detalhe/subdetalhe, etc;
d) formatação das páginas em colunas;
e) depuração do código do relatório;
f) filtros de exportação para diferentes formatos de arquivo – pdf, xls, html, jpeg, etc.
g) visualização de miniatura de páginas na pré-visualização do relatório;
h) geração de relatórios multicolunas (*cross-tab*);
i) não necessita distribuir bibliotecas (DLLs) juntamente com o aplicativo;
j) acesso aos demais objetos contidos na aplicação.

Dentre os principais motivos para a escolha deste gerador de relatório estão a

[2] Programa com funcionalidades disponíveis por determinado período de tempo.

possibilidade de criação de relatórios multicolunas, o que é essencial para apresentar os resultados do fluxo de caixa; a sua fácil integração com o Delphi em função de ser um pacote de componentes adicionado à biblioteca; a maneira WYSIWYG[3] de preparar e criar o *layout* do relatório.

[3] What You See Is What You Get - o que você vê é o que você tem. Permite que um documento seja manipulado na tela com a mesma aparência de sua utilização final.

5 RESULTADOS OBTIDOS

Neste capítulo são apresentados os resultados obtidos com o desenvolvimento do aplicativo. Como mencionado anteriormente, o aplicativo foi subdividido em incrementos, sendo que cada incremento representa uma parte (ou módulo) específica do aplicativo.

Ao carregar o aplicativo, o usuário tem acesso à janela de *login*, onde deve informar o servidor que deseja conectar, o usuário e a senha, conforme ilustrado na Figura 28.

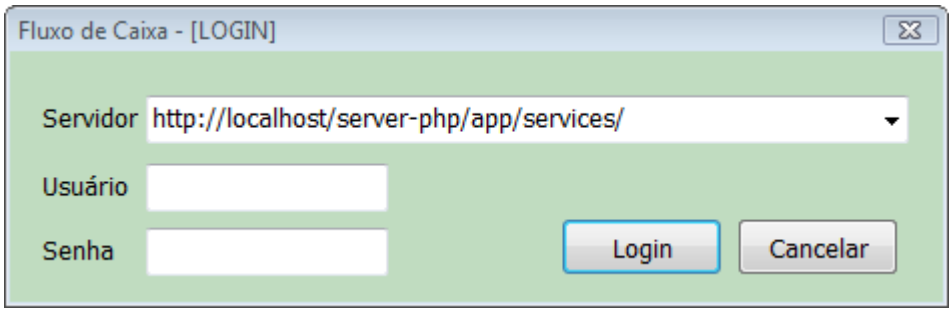


Figura 28 – Janela de login

Ao efetuar a autenticação no sistema, o aplicativo realiza uma consulta no serviço *Web* do servidor especificado, e valida o usuário e senha informados. Se os dados estiverem corretos o usuário é autenticado e a janela com o menu principal é carregada, conforme visualizado na Figura 29.

Figura 29 – Menu principal do aplicativo

5.1 Cadastros

De uma forma geral, todos os cadastros têm o mesmo padrão e comportamento. Ao abrir um dos cadastros disponíveis, a janela aberta terá o mesmo padrão visual para usuários, tipos de entidades, categorias, entidades e lançamentos. A janela do cadastro é subdividida em duas abas: grade e formulário. Na aba “grade” são visualizados os registros do respectivo cadastro em forma de lista, possibilitando que o usuário realize a ordenação alfabética pelo campo desejado, apenas clicando no título da coluna, conforme ilustrado na Figura 30. Tanto na aba “grade” quanto “formulário” o usuário poderá realizar modificações no cadastro, com a diferença que estando em “formulário” será visualizado um único registro por vez.

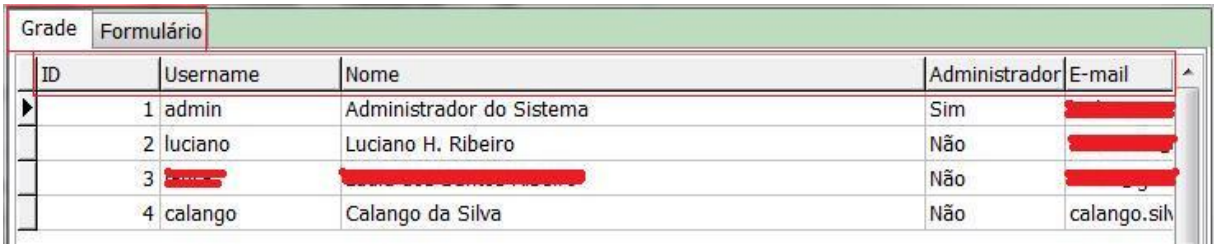


Figura 30 – Barra superior dos cadastros

Na parte inferior da janela dos cadastros está disponível a barra de ações e também um painel indicando o estado atual e a quantidade de registros cadastrada, conforme visível na Figura 31, identificando as seguintes ações:

a) anterior, posiciona a visualização no registro anterior;
b) próximo, posiciona a visualização no próximo registro;
c) inserir, coloca o cadastro em modo de inclusão de novo registro;
d) excluir, exclui o registro visualizado do cadastro;
e) alterar, coloca o cadastro em modo de alteração de registro;
f) gravar, salva em memória o novo registro ou o registro alterado;
g) cancelar, cancela a inclusão ou alteração do registro em memória;
h) atualizar, recarrega os registros do cadastro por intermédio de uma chamada ao respectivo serviço *Web*;
i) aplicar, salva todas as modificações realizadas em memória para a base de dados, submetendo os registros através do serviço *Web* específico;

j) desfazer, cancela as modificações feitas em memória, realizando-as uma a uma.

Figura 31 – Barra de ações da janela de cadastros

Como constatado através da barra de ações, as modificações (inclusões, alterações e exclusões) são realizadas, em um primeiro momento, em memória. Desta forma o usuário pode proceder com várias modificações e, no momento que desejar, efetivar a sua gravação na base de dados, invocando o serviço *Web* que irá se encarregar de persistir os dados.

5.1.1 Usuários

No cadastro de usuários (Figura 32) estão contidos os usuários que têm permissão de acesso ao sistema. A única distinção entre um usuário administrador e um comum, é que o usuário administrador tem permissão de acesso completo ao sistema e um usuário comum (não administrador) tem permissão de acesso somente aos lançamentos e ao fluxo de caixa.

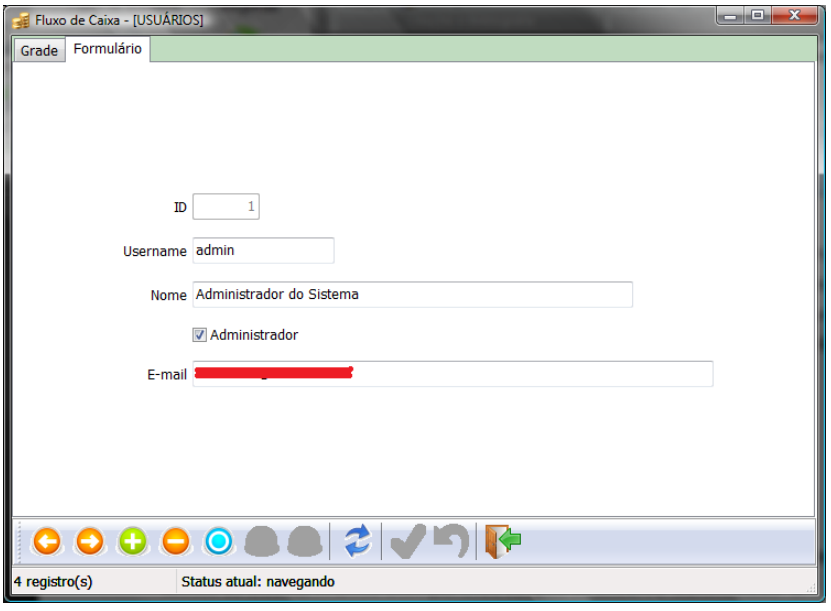


Figura 32 – Tela do cadastro de usuários

5.1.2 Tipos de entidade

Os tipos de entidade (Figura 33) são utilizados para realizar a classificação nas entidades do sistema, tais como: fornecedores, clientes, transportadores, etc.

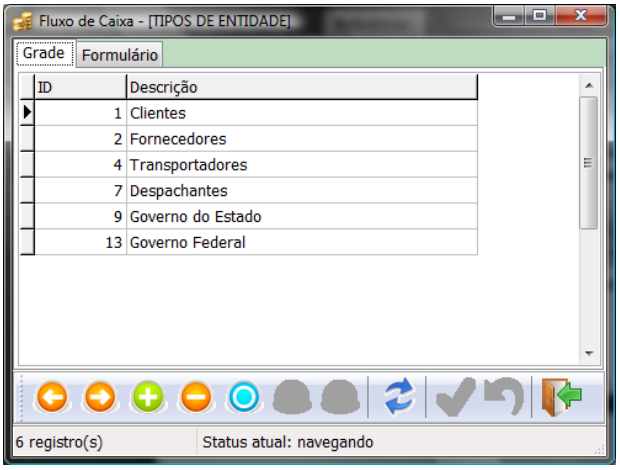


Figura 33 – Tela do cadastro de tipos de entidade

5.1.3 Categorias

O cadastro de categorias (Figura 34) é utilizado dentro do sistema para classificar os lançamentos em grupos de contas. As categorias estão dispostas em uma estrutura de árvore, ou seja, para cada nova categoria é possível especificar a qual categoria ela pertence, mantendo assim uma hierarquia de categorias.

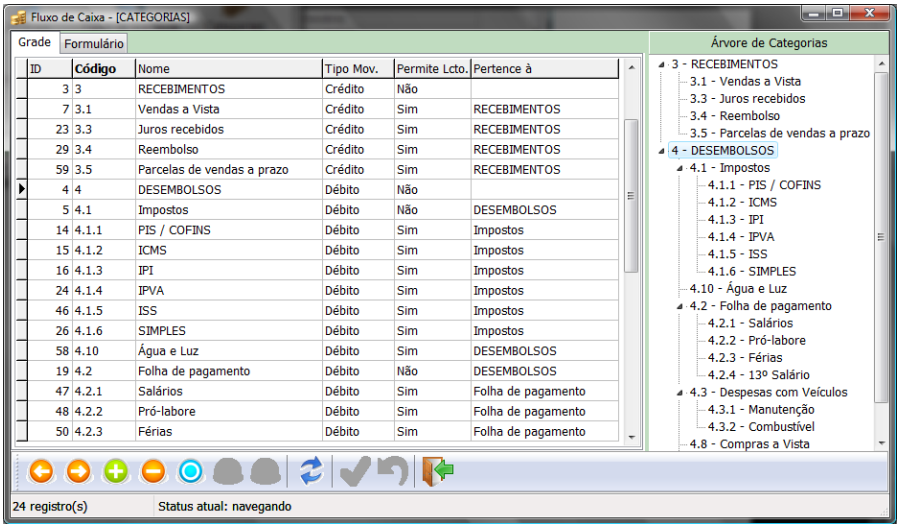


Figura 34 – Tela do cadastro de categorias

Cada categoria possui um campo código que é gerado automaticamente pelo sistema (serviço *Web*), de acordo com o nível e a categoria à que pertence o registro corrente. O sistema de fluxo de caixa permite que sejam mantidos três níveis de categorias e restringe que o tipo de movimentação (crédito ou débito) deva ser o mesmo da categoria à qual ela pertence. O campo “Permite Lcto.” é utilizado para que permita que sejam atribuídos lançamentos somente às categorias que possuírem este campo marcado como “Sim”, evitando que sejam atribuídos níveis de categorias indevidos aos lançamentos.

5.1.4 Entidades

Este cadastro é utilizado para armazenar as empresas e pessoas utilizadas no fluxo de caixa. Os registros são subdivididos em tipos para caracterizar as entidades em clientes, fornecedores, transportadores, etc. Na Figura 35 é exibido um exemplo das entidades.

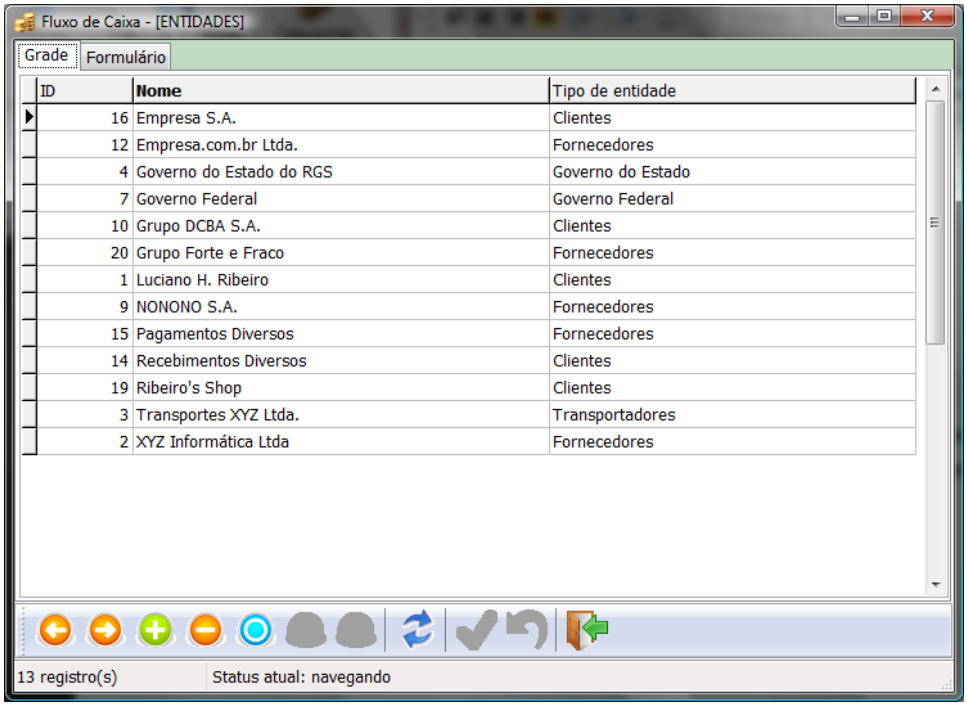
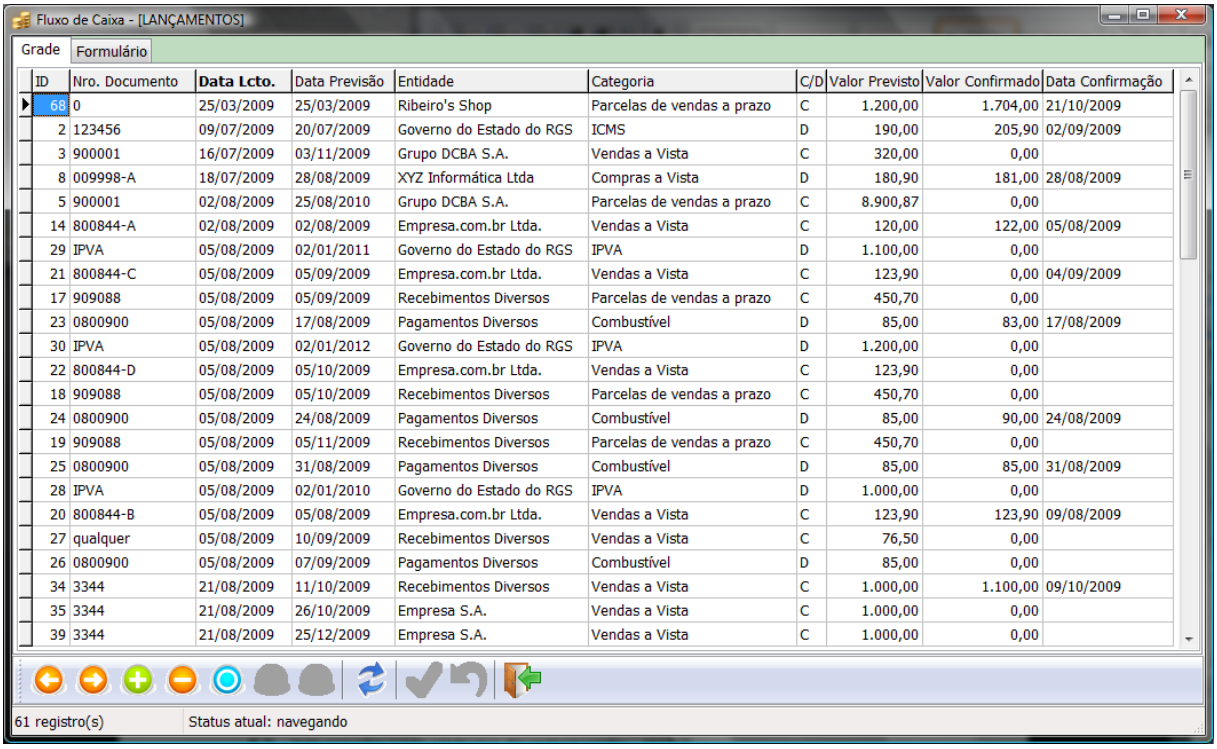


Figura 35 – Tela do cadastro de entidades

5.1.5 Lançamentos

Os lançamentos são os registros que compõe a base para o fluxo de caixa. São compostos por movimentações de crédito e débito, necessitando que cada registro seja composto por número, entidade, categoria, valor previsto e valor confirmado. Além dos campos para controle de datas: de lançamento, de previsão e de confirmação, conforme ilustrado na Figura 36.



ID	Nro. Documento	Data Lcto.	Data Previsão	Entidade	Categoria	C/D	Valor Previsto	Valor Confirmado	Data Confirmação
68	0	25/03/2009	25/03/2009	Ribeiro's Shop	Parcelas de vendas a prazo	C	1.200,00	1.704,00	21/10/2009
2	123456	09/07/2009	20/07/2009	Governo do Estado do RGS	ICMS	D	190,00	205,90	02/09/2009
3	900001	16/07/2009	03/11/2009	Grupo DCBA S.A.	Vendas a Vista	C	320,00	0,00	
8	009998-A	18/07/2009	28/08/2009	XYZ Informática Ltda	Compras a Vista	D	180,90	181,00	28/08/2009
5	900001	02/08/2009	25/08/2010	Grupo DCBA S.A.	Parcelas de vendas a prazo	C	8.900,87	0,00	
14	800844-A	02/08/2009	02/08/2009	Empresa.com.br Ltda.	Vendas a Vista	C	120,00	122,00	05/08/2009
29	IPVA	05/08/2009	02/01/2011	Governo do Estado do RGS	IPVA	D	1.100,00	0,00	
21	800844-C	05/08/2009	05/09/2009	Empresa.com.br Ltda.	Vendas a Vista	C	123,90	0,00	04/09/2009
17	909088	05/08/2009	05/09/2009	Recebimentos Diversos	Parcelas de vendas a prazo	C	450,70	0,00	
23	0800900	05/08/2009	17/08/2009	Pagamentos Diversos	Combustível	D	85,00	83,00	17/08/2009
30	IPVA	05/08/2009	02/01/2012	Governo do Estado do RGS	IPVA	D	1.200,00	0,00	
22	800844-D	05/08/2009	05/10/2009	Empresa.com.br Ltda.	Vendas a Vista	C	123,90	0,00	
18	909088	05/08/2009	05/10/2009	Recebimentos Diversos	Parcelas de vendas a prazo	C	450,70	0,00	
24	0800900	05/08/2009	24/08/2009	Pagamentos Diversos	Combustível	D	85,00	90,00	24/08/2009
19	909088	05/08/2009	05/11/2009	Recebimentos Diversos	Parcelas de vendas a prazo	C	450,70	0,00	
25	0800900	05/08/2009	31/08/2009	Pagamentos Diversos	Combustível	D	85,00	85,00	31/08/2009
28	IPVA	05/08/2009	02/01/2010	Governo do Estado do RGS	IPVA	D	1.000,00	0,00	
20	800844-B	05/08/2009	05/08/2009	Empresa.com.br Ltda.	Vendas a Vista	C	123,90	123,90	09/08/2009
27	qualquer	05/08/2009	10/09/2009	Recebimentos Diversos	Vendas a Vista	C	76,50	0,00	
26	0800900	05/08/2009	07/09/2009	Pagamentos Diversos	Combustível	D	85,00	0,00	
34	3344	21/08/2009	11/10/2009	Recebimentos Diversos	Vendas a Vista	C	1.000,00	1.100,00	09/10/2009
35	3344	21/08/2009	26/10/2009	Empresa S.A.	Vendas a Vista	C	1.000,00	0,00	
39	3344	21/08/2009	25/12/2009	Empresa S.A.	Vendas a Vista	C	1.000,00	0,00	

Figura 36 – Tela do cadastro de lançamentos em grade

Para a inclusão de um novo lançamento é exigido que as datas de previsão e confirmação (quando informada) sejam iguais ou superiores que a data do lançamento (data de inclusão do registro). Além disto, a lista de categorias é restrita às categorias que permitem lançamentos. É possível, durante a inclusão, que o mesmo registro seja repetido automaticamente por um determinado período e por uma quantidade de vezes definida durante a inclusão, visualizado, a seguir, na Figura 37.

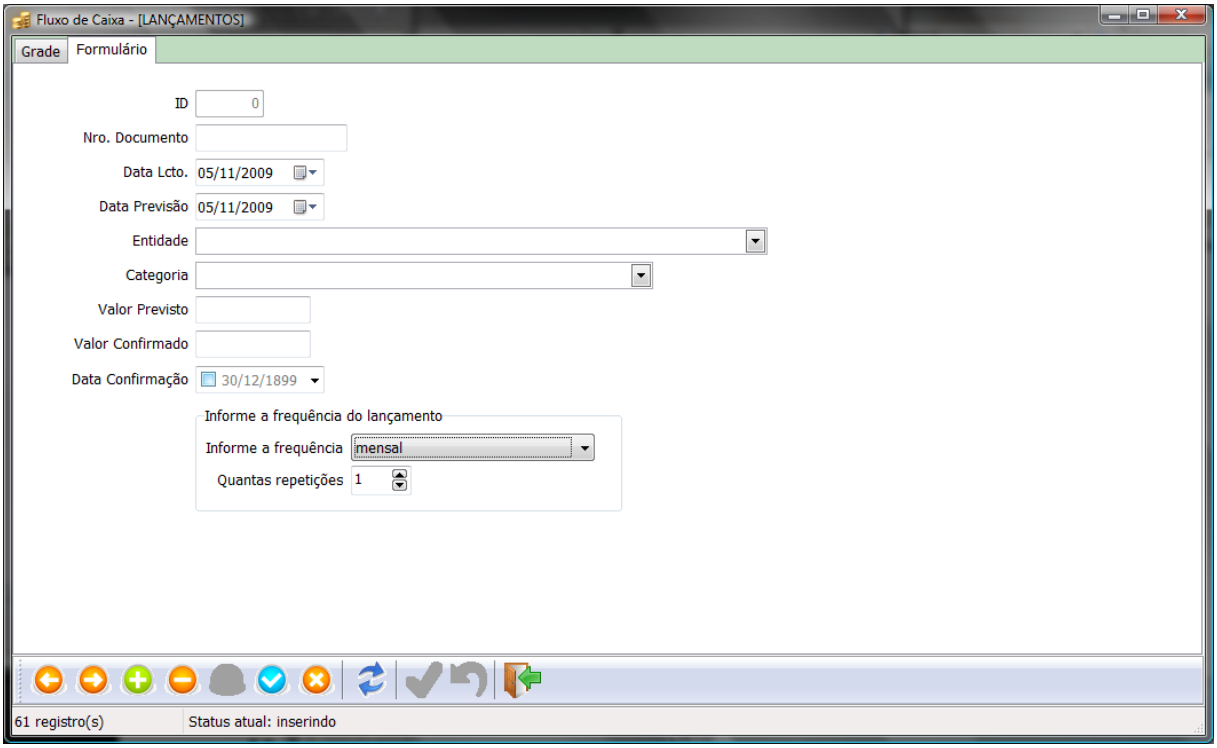


Figura 37 – Tela do cadastro de lançamentos por formulário

5.2 Fluxo de caixa

O fluxo de caixa pode ser visualizado em forma de cubo de decisão e relatório. Para a visualização dos dados é necessário que sejam especificados o período utilizado para a seleção, o intervalo de datas desejado e a forma que o usuário deseja que os dados sejam agrupados. A Figura 38 identifica os campos necessários.

Figura 38 – Tela de seleção do fluxo de caixa

a) no campo “selecionar por período” deve ser especificada qual a data utilizada no intervalo de seleção, com as opções: data de lançamento, data de previsão e data de confirmação. Quando não especificado, o aplicativo irá assumir por padrão a data de previsão;
b) nos campos ao lado do período, deve ser especificado o intervalo de datas que o usuário desejar para carregar os lançamentos. Por padrão,

serão assumidos o primeiro e o último dia do ano corrente;

c) o campo “agrupar de forma” é utilizado para ser especificada a forma que o usuário deseja que os dados sejam agrupados: diário, semanal, mensal, trimestral e anual. Por padrão, o aplicativo sugere o agrupamento mensal.

Após o usuário informar os campos de seleção, é preciso que seja indicada qual a forma de exibição dos dados, de acordo com a aba selecionada para isto.

5.2.1 Cubo de decisão

Para exibir os dados em forma de cubo, o usuário necessita apenas acionar o ícone específico para carregar os lançamentos, conforme a seleção realizada, que as movimentações serão visualizadas. As principais funcionalidades estão identificadas a seguir e ilustradas na Figura 39.

a) carrega os registros de movimentação de acordo com a seleção realizada;
b) após visualizados os dados, a funcionalidade de impressão do relatório idêntico ao visualizado na tela;
c) área com as dimensões (campos) utilizadas nas linhas do cubo;
d) área com as dimensões (campos) utilizadas nas colunas;
e) área com as dimensões não utilizadas em linhas ou colunas;
f) área com os totalizadores (cálculos) utilizados;
g) botão utilizado para aumentar (*drill down*) ou diminuir (*drill up*) o nível de detalhamento dos dados;
h) botão usado para exibir o gráfico com os valores relativos ao respectivo totalizador.

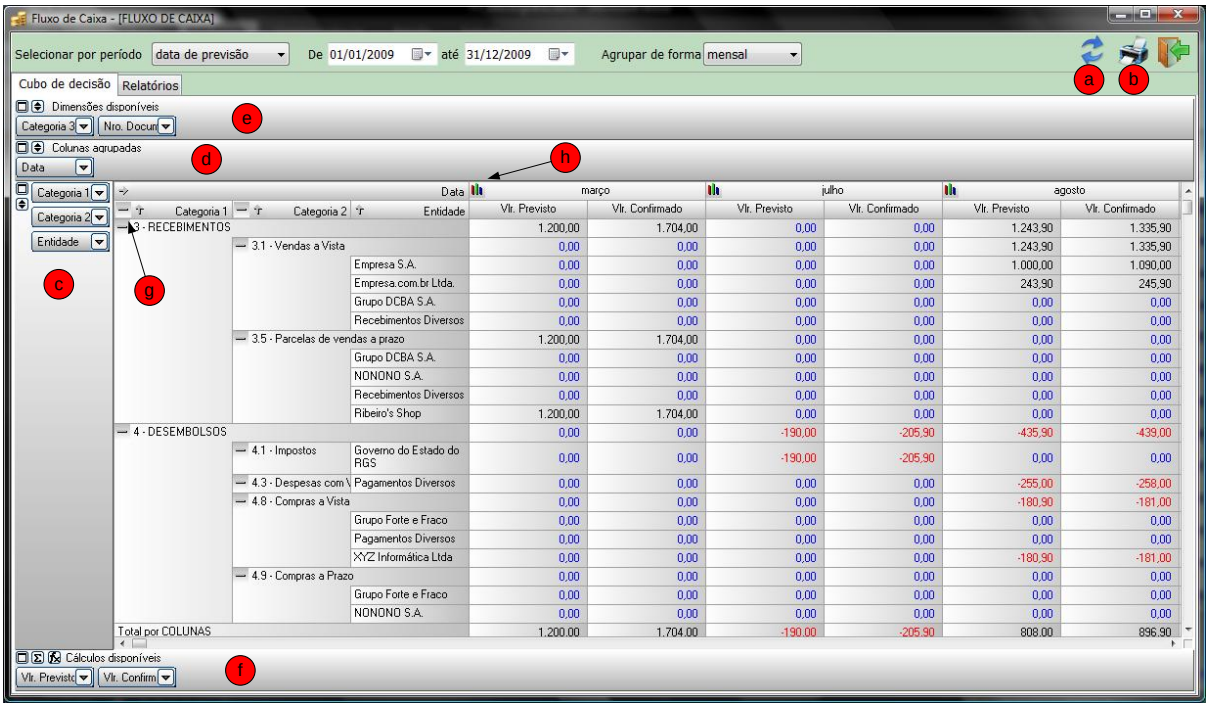


Figura 39 – Tela do cubo de decisão

As áreas com as dimensões de colunas (d), linhas (c) e dimensões disponíveis (e) possibilitam que os campos sejam movidos (arrastar e soltar) para modificar a visualização e disposição dos dados dentro do cubo, assim como, a ordem em que os campos são visualizados dentro da mesma área.

Ao acionar o botão para exibir gráfico (h), abre-se uma nova janela com a visualização do gráfico relativo ao totalizador, com as funcionalidades peculiares.

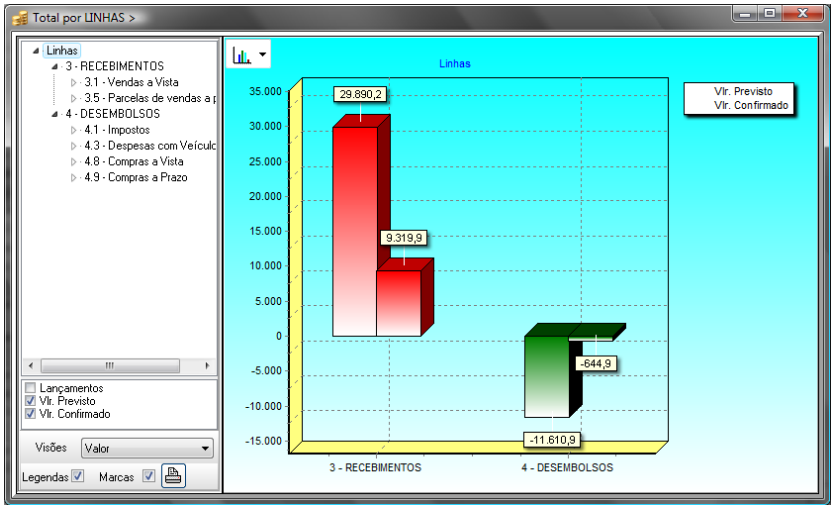


Figura 40 – Tela com o gráfico do totalizador

Os dados do cubo também podem ser vistos em forma de relatório (Figura 41), idêntico à visualização corrente, quanto a agrupamento, nível de detalhamento, linhas, colunas e totalizadores.

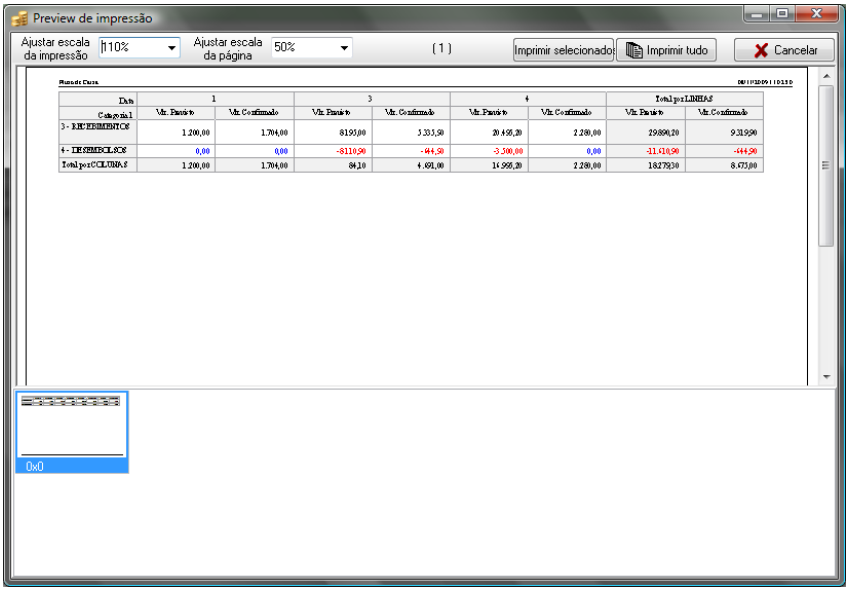


Figura 41 – Tela com o relatório do cubo de decisão

5.2.2 Relatório

O relatório do fluxo de caixa possui opções de seleção de dados específicos, que são complementares às opções gerais do fluxo de caixa, onde o usuário pode especificar um saldo inicial de caixa e os campos e valores que deseja visualizar na impressão, conforme a Figura 42.

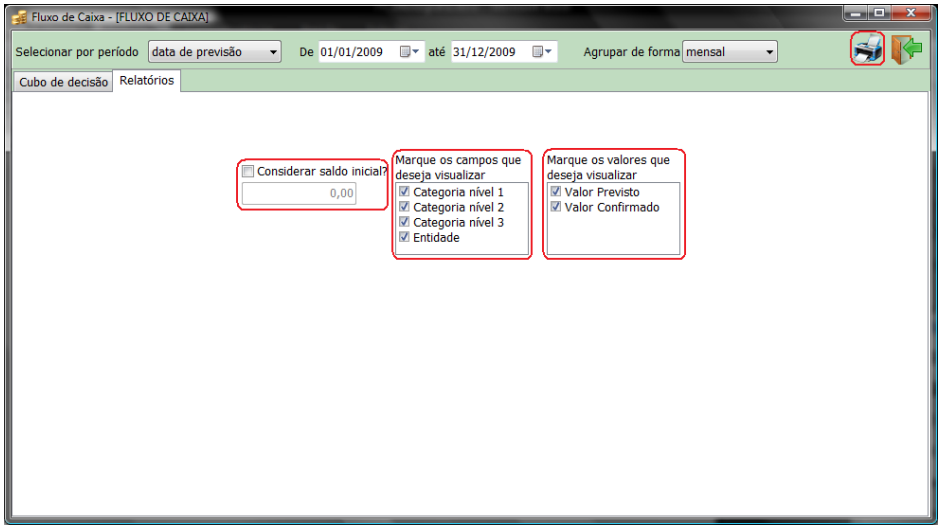


Figura 42 – Tela de seleção de relatório

Depois de aplicados os critérios de seleção, o usuário aciona o ícone de visualização do relatório exibindo uma pré-visualização com suas funcionalidades

específicas e ilustrado na Figura 43.

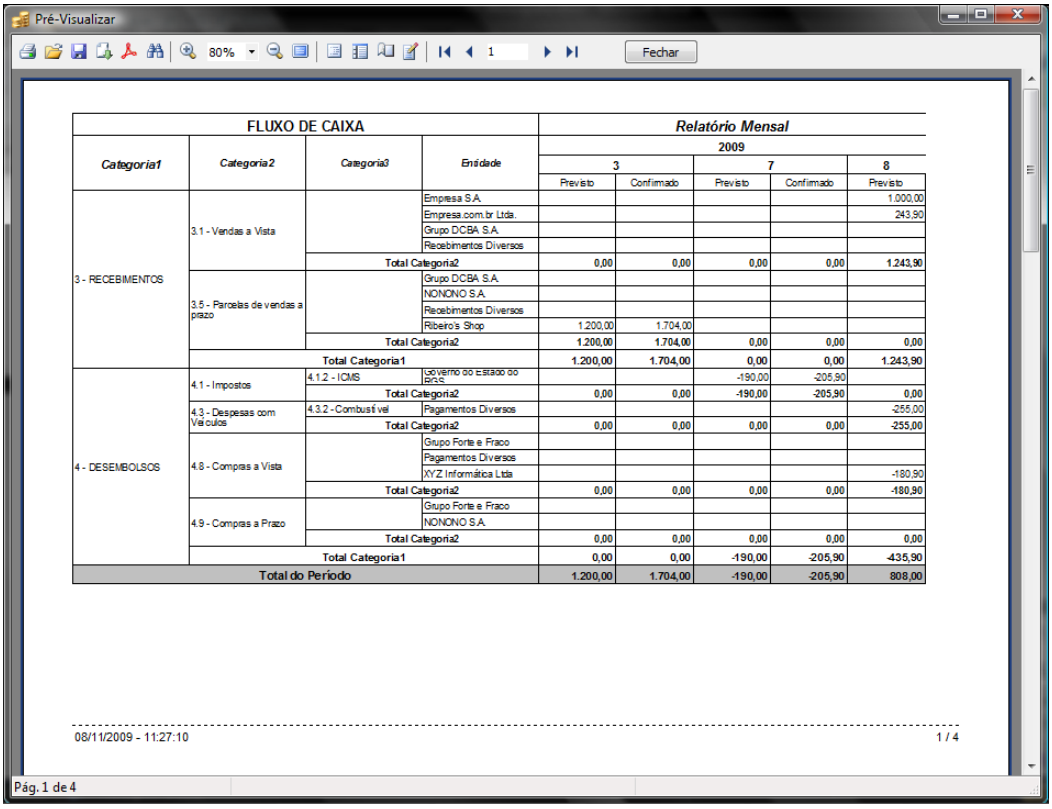


Figura 43 – Tela de pré-visualização de relatório

5.3 Importar dados

A importação de dados consiste em carregar a movimentação a partir de um sistema de retaguarda. Esta importação é realizada pelo serviço *Web* que carrega os dados a partir de arquivos XML que devem estar formatados a partir de esquemas pré-definidos no servidor, com possibilidade de importar dados para o cadastro de entidades, categorias e lançamentos.

Para a importação dos dados de retaguarda foram realizados testes com o sistema Hábil Empresarial, que possui uma versão gratuita, possibilitando o lançamento da movimentação de contas a pagar e receber, que são a base do fluxo de caixa. Neste caso, foi realizada a engenharia reversa da base de dados do Hábil para identificar as tabelas necessárias, resultado no mapeamento de tabelas e campos conforme o Quadro 25, ilustrando esta situação.

HÁBIL				FLUXO DE CAIXA			
Tabela	**Caixa_Contas**			Tabela	**tb_categorias**		
	Campo	**Tipo**	**Tamanho**		**Campo**	**Tipo**	**Tamanho**
(PK)	Codigo	Número					
(PK)	Empresa	Número					
				(PK)	id_categoria	integer	
	Descrição	Texto	80		nome	varchar	40
(FK)	Código_Conta	Texto	30		codigo	varchar	20
	Permite_Lançamentos	Sim/Não			permite_lcto	char	1
					tipo_mov	char	1
					nivel	smallint	
Tabela	**Cliente**			Tabela	**tb_entidades**		
	Campo	**Tipo**	**Tamanho**		**Campo**	**Tipo**	**Tamanho**
(PK)	Codigo	Número					
(PK)	Empresa	Número					
				(PK)	id_entidade	integer	
	Cliente_Nome	Texto	50		nome	varchar	60
	Cliente_Razão_Social	Texto	50				
				(FK)	id_tipo	integer	
Tabela	**Fornecedor**			Tabela	**tb_entidades**		
	Campo	**Tipo**	**Tamanho**		**Campo**	**Tipo**	**Tamanho**
(PK)	Codigo	Número					
(PK)	Empresa	Número					
				(PK)	id_entidade	integer	
	Razão_Social	Texto	50		nome	varchar	60
	Nome_Fantasia	Texto	50				
				(FK)	id_tipo	integer	
Tabela	**Contas_Receber**			Tabela	**tb_lancamentos**		
	Campo	**Tipo**	**Tamanho**		**Campo**	**Tipo**	**Tamanho**
(PK)	Codigo	Número					
(PK)	Empresa	Número					
				(PK)	id_lcto	integer	
	Documento	Texto	25		nr_documento	varchar	15
	Código_Cliente	Número					
					id_entidade	integer	
	Data_Conta	Data/Hora			dt_lcto	datetime	
	Data_Vencimento	Data/Hora			dt_previsao	datetime	
	Data_Pagamento	Data/Hora			dt_confirmacao	datetime	
	Valor	Número			vl_previsto	float	
	Juros	Número					
	Desconto	Número					
	Deduções	Número					
	Acréscimos	Número					
	Total	Número			vl_confirmado		
(FK)	Conta_Caixa	Texto	30				
				(FK)	id_categoria	integer	
Tabela	**Contas_Pagar**			Tabela	**tb_lancamentos**		
	Campo	**Tipo**	**Tamanho**		**Campo**	**Tipo**	**Tamanho**
(PK)	Codigo	Número					
(PK)	Empresa	Número					
				(PK)	id_lcto	integer	
	Documento	Texto	25		nr_documento	varchar	15
	Código_Fornecedor	Número					
					id_entidade	integer	
	Data_Conta	Data/Hora			dt_lcto	datetime	
	Data_Vencimento	Data/Hora			dt_previsao	datetime	
	Data_Pagamento	Data/Hora			dt_confirmacao	datetime	
	Valor	Número			vl_previsto	float	
	Juros	Número					
	Desconto	Número					
	Deduções	Número					
	Multa	Número					
	Total	Número			vl_confirmado		
(FK)	Conta_Caixa	Texto	30				
				(FK)	id_categoria	integer	

Quadro 25 – Mapeamento para importação de dados do Hábil

Depois de realizado o mapeamento, o serviço *Web* específico comunica com a base do Hábil e gera os arquivos XML atendendo as especificações dos esquemas de dados e os retorna para a *interface* de importação para que o usuário passa realizar as manipulações necessárias antes de salvar os dados nos respectivos cadastros.

A janela de importação de dados possibilita que o usuário especifique qual tabela deseja carregar os registros. Neste momento o aplicativo cliente irá requisitar os registros ao serviço *Web* e, depois de exibidos, será indicada a situação de cada um, informando se o mesmo já existe (importado anteriormente) ou é um registro novo.

Para cada registro, o usuário poderá especificar uma ação, ou seja, se deseja importar ou ignorar a importação do registro. Após atribuídas as ações desejadas, o usuário pode realizar a importação, submetendo-os ao serviço *Web* que irá se encarregar de salvar as informações na tabela especificada, conforme indicado na Figura 44.

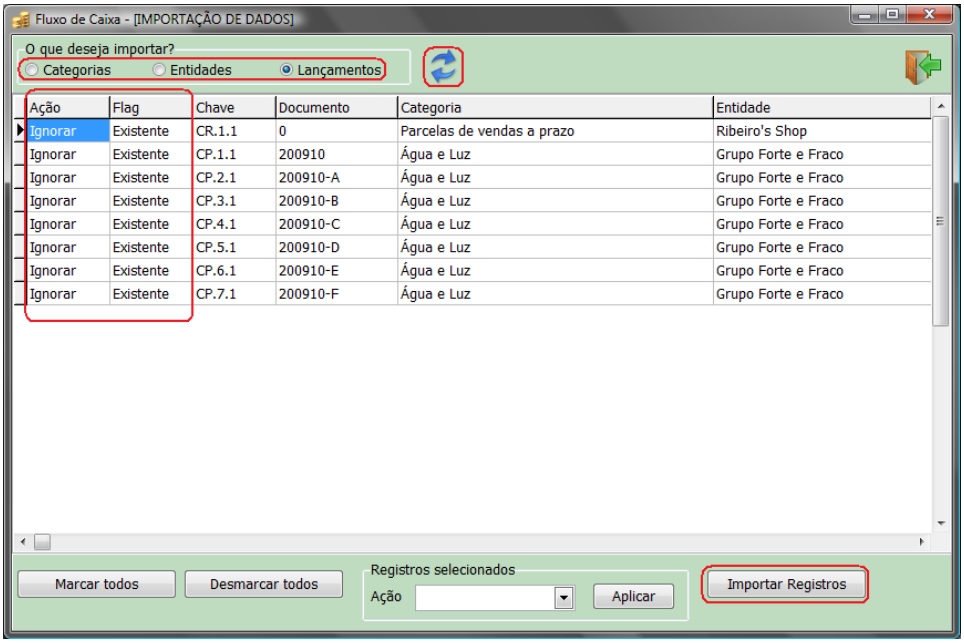


Figura 44 – Tela de importação de dados

6 CONCLUSÃO

No presente trabalho foi apresentada uma solução de fluxo de caixa para auxiliar os gestores financeiros na análise das informações e gestão do negócio da empresa. O fluxo de caixa, neste sentido, serve como ferramenta de grande valia, para que o administrador possa tomar as decisões corretas e realizar os ajustes nos recursos financeiros da empresa no tempo correto, prevenindo-se de situações que possam gerar dificuldades dentro da organização ou, então, possibilitando a aplicação de recursos que estejam à disposição e evitar o desperdício de valores que poderiam estar aplicados ou que investimentos possam ser realizados em um determinado período.

O *software* desenvolvido não tem por finalidade ser um sistema integrado de gestão (ERP), mas sim ser utilizado em conjunto com sistemas de retaguarda para contribuir com a visualização e análise dos resultados financeiros da empresa. Neste aspecto, a utilização de *Web Services* e da arquitetura orientada a serviços facilitou a integração entre diferentes sistemas, permitindo que as informações existentes em módulos financeiros sejam utilizadas dentro do fluxo de caixa.

Outro aspecto que justificou a utilização de *web services* está no fato de permitir que o fluxo de caixa seja acessado remotamente, sem a necessidade que o aplicativo seja executado no mesmo ambiente de rede da empresa, mas também acessando os dados a distância, por intermédio de conexão com a *Internet*. Para atender a essa situação é preciso que o gestor financeiro tenha instalado em seu computador apenas o aplicativo cliente do fluxo de caixa, sem a necessidade de implantação de todo o ambiente, como servidor *Web*, banco de dados, etc. Neste caso, a tecnologia empregada permite a integração entre diferentes plataformas, pois não restringe o uso do sistema em um ambiente uniforme, possibilitando sua implantação em um provedor de *Internet*, caso não exista uma estrutura de servidor *Web* apropriada.

A facilidade de realizar os lançamentos de créditos e débitos diretamente no aplicativo agiliza o processo de análise da informação, pois não restringe que os lançamentos sejam realizados somente no sistema de retaguarda e depois disto carregados para o fluxo de caixa e avaliados. Além disto, o aplicativo de fluxo de caixa pode ser executado isoladamente, sem a necessidade de estar acoplado a

outro sistema para que seus dados sejam alimentados.

A utilização de cubo de decisão na análise dos dados apresentados oferece uma maneira ágil de manusear os dados, permitindo uma análise sintética dos recursos financeiros e também uma avaliação analítica, permitindo a visualização dos dados no nível mais baixo de detalhamento. Outra vantagem identificada na utilização de cubo de decisão está na possibilidade de modificar a forma com que a informação é apresentada, alternando entre os níveis de agrupamento e de detalhamento.

A forma com que as informações são apresentadas auxilia o processo de análise dos dados, pois apresenta em uma ferramenta OLAP para a visão financeira da empresa. Sendo assim, quem estiver operando o sistema, pode alternar entre diferentes perspectivas e expandir o nível de detalhamento em um único local, sem a necessidade de abrir janelas simultâneas para realizar os comparativos ou buscar o detalhamento das informações, algo que é muito comum em grande parte dos *softwares* de gestão.

O aplicativo de fluxo de caixa implementado se apresenta como uma alternativa aos administradores financeiros que desejam a análise das informações previstas e realizadas, pois permite que elas sejam comparadas em diferentes períodos e em formas de visualização diferentes, mas não pode ser considerado como uma solução definitiva para todas as necessidades dos gestores, pois, como todo *software*, deve estar em constante melhoria para cada vez mais se aproximar de uma solução ideal de análise dos recursos financeiros. De qualquer maneira, os resultados apresentados demonstram que os objetivos do presente trabalho foram atingidos.

REFERÊNCIAS

BOOCH, Grady; RUMBAUGH, James; JACOBSON, Ivar; **UML,** guia do usuário. Rio de Janeiro: Campus, 2000.

Borland Delphi 7 Developer's Guide. Borland Software Corporation. Scotts Valley, EUA. 2002.

CANTU, Marco. **Dominando o Delphi 7**: A Bíblia. São Paulo: Pearson Makron Books, 2003.

DALL'OGLIO, Pablo. **PHP**: programando com orientação a objetos. São Paulo: Novatec, 2007.

ERL, Thomas. **Service-Oriented Architecture** – A Field Guide to Integrating XML and Web Services. New Jersey: Prentice Hall – PTR, 2004.

FastReport VCL. Report generator for Delphi – Fast Reports Inc. Disponível em: <http://fast-report.com/en/products/report-generator-for-delphi-fastreport-4.html>. Acesso: 22 set. 2009.

HTTP Server Project. Apache. Disponível em: <http://httpd.apache.org>. Acesso: 19 set. 2009.

JOSUTTIS, Nicolai M. **SOA na prática**. Rio de Janeiro: Alta Books, 2008.

KOSCIANSKI, André; SOARES, Michel dos S. **Qualidade de software**. 2 ed. São Paulo: Novatec, 2007.

MEDEIROS, Ernani. **Desenvolvendo software com UML 2.0** – Definitivo. São Paulo: Pearson Makron Books, 2004.

MELLO, R. S. **Gerenciamento de dados XML** In: V Escola de Informática Norte da SBC, 2003, Palmas. ENCOINFO/EIN 2003. Palmas: Centro Universitário Luterano de Palmas - ULBRA, 2003. v.1. p.15 – 34.

MINETTO, Elton L. **Frameworks para desenvolvimento em PHP**. São Paulo: Novatec, 2007.

MySQL. Disponível em: <http://www.mysql.com/why-mysql>. Acesso: 19 set. 2009.

NASCIMENTO, Daniel. SOA Arquitetura Orientada a Serviços: Será um novo Paradigma?. Active Delphi, ano III, ed. 36, p. 16-17, fev. 2007.

NETTO, Eduardo José. **Olho no caixa!**: Como desenvolver Sua Visão sobre a Administração Financeira. Nobel, 1999. Disponível em: <http://books.google.com.br/books?id=wsSV136lQJUC>. Acesso: 27 out. 2008.

NuSOAP - SOAP Toolkit for PHP. Disponível em: < http://nusoap.sourceforge.net/>. Acesso: 29 ago 2009.

PAMPLONA, Vitor F. **Web Services**. Construindo, disponibilizando e acessando Web Services via J2SE e J2ME. Disponível em: <http://www.javafree.org/content/view.jf?idContent=4>. Acesso: 16 nov. 2008.

PHP: Manual do PHP. PHP Documentation Group. Disponível em: <http://www.php.net/manual/pt_BR/>. Acesso em: 27 ago. 2009.

PivotCube VCL. PivotCube. Disponível em: <http://www.pivotcube.com>. Acesso: 22 set. 2009.

PRESSMAN, Roger S. **Engenharia de Software**. 6 ed. Rio de Janeiro: McGraw-Hill, 2006.

Project Homepage. Microsoft Office Online. Disponível em: <http://office.microsoft.com/pt-br/project/default.aspx>. Acesso: 07 set. 2009.

Propel. Propel PHP Project. Disponível em: <http://propel.phpdb.org/trac>. Acesso: 19 set. 2009.

REZENDE, Denis Alcides. **Engenharia de software e sistemas de informação**. 3 ed. Rio de Janeiro: Brasport, 2005.

ROSS, Stephen A.; WESTERFIELD, Randolph W.; JORDAN, Bradford D. **Princípios de administração financeira**. 2 ed. São Paulo: Atlas, 2000.

SCOTT, Kendall. **O processo unificado explicado**. Porto Alegre: Bookman, 2003.

SILBERSCHATZ, Abraham; KORTH, Henry F.; SUDARSHAN, S**. Sistemas de banco de dados**. 3 ed. São Paulo: Makron Books, 1999.

SOMMERVILLE, Ian. **Engenharia de software**. 6 ed. São Paulo: Addison Wesley, 2003.

W3C – World Wide Web Consortium. Disponível em: <http://www.w3.org/>. Acesso: 31 ago. 2009.

W3Schools – SOAP Tutorial. Disponível em: <http://www.w3schools.com/SOAP>. Acesso: 31 ago. 2009.

ZDANOWICZ, José Eduardo. **Fluxo de caixa**: uma decisão de planejamento e controles financeiros. 7 ed. Porto Alegre: Sagra, 1998.