UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

ESCOLA POLITÉCNICA

DEPARTAMENTO DE ENGENHARIA ELETRÔNICA E DE COMPUTAÇÃO

SISTEMA DE CONTROLE FINANCEIRO PESSOAL
PRP – Personal Resource Planning

Autor: **Marcelo Vicente Vianna Magalhães**

Orientador 1: **Antônio Cláudio Gómez de Sousa**

Orientador 2: **Sergio Palma da Justa Medeiros**

Orientador 3: **Sérgio Barbosa Villas-Boas**

DEL
Dezembro de 2009

A uma pessoa que me mostrou o outro lado da vida,
uma vida com alegrias, tristezas, problemas e superações,
contudo com muito companheirismo e amor.
A minha esposa.

AGRADECIMENTOS

Gostaria de agradecer a todos os docentes do Departamento de Engenharia Eletrônica que sempre estiveram apostos para ajudar nas duvidas surgidas durante minha estada na academia e principalmente na execução deste projeto, sem ajuda de tais professores não seria possível completar tal feito com a maximização do aprendizado.

Durante a execução aprendi com nossos mestres, e até algumas vezes os ensinei. É nesta interatividade entre nós, alunos e professores, que são construídos os alicerces de um novo profissional que tem como vantagem a formação em uma das melhores universidades do Brasil.

Um agradecimento especial faz-se necessário aos meus orientadores que ao fim deste processo passaram para o nível de colegas de profissão e amigos, me auxiliando e esclarecendo dúvidas durante a vida deste projeto.

Por fim, contudo não menos importante de todos os agradecimentos, ao Povo Brasileiro e nossos governantes que me deram a oportunidade de estudar em uma das mais conceituadas universidade das Américas, a Universidade Federal do Rio de Janeiro.

Resumo

Este projeto de graduação tem a finalidade de colocar em prática todos os conhecimentos adquiridos por mim durante a minha estada no curso de Engenharia Eletrônica da UFRJ. O projeto consiste no desenvolvimento de um sistema de computador voltado para a área financeira, mais precisamente um gerenciador de finanças. O sistema foi modelado e projetado utilizando-se as mais modernas técnicas de projeto utilizadas na UML, linguagem de modelagem de sistemas de computador. Para a implementação foram utilizadas diversas bibliotecas para prover acesso a dados, interface gráfica com o usuário, interface entre sistemas, bem como a linguagem de programação C++, na qual todo o sistema foi codificado.

Outra finalidade do desenvolver deste projeto foi, além de aplicar os conhecimentos em um produto final, o de conciliar os ensinamentos de engenharia para serem aplicados em outras áreas, no caso a financeira. Ao longo deste projeto o leitor verá termos técnicos de engenharia, de finanças, contabilidade, enfim uma riqueza de conhecimentos unidos para um objetivo comum.

Abstract

This project of graduation has the purpose to place in practical all the knowledge acquired for me during my sojourn in the course of Electronic Engineering of the UFRJ. The project consists of the development of a system of computer directed toward the financial area, more necessarily a finances manager. The system was shaped and projected using the most modern used techniques of project in the UML, language of modeling of computer systems. For the implementation the data had been used diverse libraries to provide access, graphical interface with the user, interface between systems, as well as the programming language C++, in which all the system was codified.

Another purpose of developing of this project was, beyond applying the knowledge in an end item, to conciliate the engineering teachings to be applied in other areas, in the case the financier. Throughout this project the reader will see terms engineering technician, of finances, accounting, at last a wealth of knowledge joined for a common objective.

Palavras-chaves

Gerenciamento financeiro

Analise financeira

Investimentos

ERP – Enterprise Resource Planning

wxWidgets

UML

Índice

Índice de gráfico e figuras

Índice de tabelas

Glossário

BMF – Bolsa Mercantil de Futuro

BOVESPA

BC

PRP SELIC

IPCA

IGP-M

Abreviaturas, Siglas, Símbolos e Sinais

RSS

XML

Feeds

1. Introdução

1.1. A necessidade

Nos últimos anos o Brasil tem passado por profundas mudanças em seus mercados financeiros, tais como *BMF*, *BOVESPA*, Corretora de Valores, Tesouro Direto, etc., visando aumentar sua penetração e entendimento junto ao povo brasileiro e até mesmo se "mostrando" mais para o mercado mundial. Estas mudanças podem ser notadas com a maior transparência nas informações, tanto por parte do governo quanto por parte das empresas e no aumento do programas de educação financeira empreitados por entidades da área, vide exemplo do Banco Central com seu programa *BC* e Universidade ou o projeto Programa Educacional Bovespa da Bovespa, entre muito outros.

Esta transparência financeira e maior facilidade de acesso às informações fizeram surgir um novo tipo de publico, o investidor de pequeno/médio porte. Esta nova classe é muito importante para a condução e aprimoramento das finanças nacionais, pois junta, tem "poder de capital" igual ou muito próximo de um mega-investidor, contudo mantendo uma maior pulverização de investimentos. A importância desta classe pode ser notada em países ditos de primeiro mundo, que dão muito valor a tais investidores, por exemplo, muitos americanos investem em ações visando longo prazo, até mesmo sua aposentadoria esta fato fez com que muitas empresas fizessem reuniões especificamente com acionistas minoritários.

Nesta reorganização dos mercados financeiros a visão do investidor de pequeno/médio porte também está sendo modificada, anteriormente tal classe tinha uma visão de curto ou curtíssimo prazo, situação esta hoje que vem se modificando sensivelmente. Esta modificação vem ocorrendo devido justamente à maior transparência financeira por parte do governo e das empresas, que publicam balanços, criam departamentos especializados em atender o pequeno e médio investidor, enfim "abrem suas finanças" para o mundo e com isso tornam-se mais atraentes aos olhos dos investidores, até mesmo despertando a curiosidade daqueles que não o eram.

É neste cenário que há uma crescente necessidade que anteriormente só era tida por grandes investidores, o de controlar as finanças pessoais. Imagine a situação de um pequeno investidor que quer saber quanto ele arrecada e gasta por mês para poder direcionar seus recursos? Quais ativos ele deve privilegiar? Onde estão as melhores opções? Qual o cenário da economia financeira mundial? Bem todas estas e outras perguntas estão hoje na cabeça dos investidores, sejam ele pequenos, médios ou grandes. E é neste contexto que o *PRP* surge, visando criar um conjunto de ferramentas para gerenciar recursos financeiros pessoais, focado especialmente no pequeno e médio investidor.

O sistema terá 3 grandes fases de implementação, que serão descritas em mais detalhes posteriormente. Uma possível divisão destas fases seria;

Fase 1 – gerenciamento de contas pessoais
Fase 2 – inteligência para auxilio na tomada de decisões
Fase 3 – conteúdo e informações financeiras em tempo real

1.2. A motivação

A motivação para tal projeto vem de uma grande ligação que tenho com mercados financeiros e conhecimentos de administração adquiridos no passado, quando fiz faculdade de Administração. Outro sim é devido ao fato que me julgo um pequeno investidor, devido a ter parte de meu sustento vindo destes ativos. Sendo assim com foi no cenário acima descrito e com minhas poucas habilidades na área financeira que resolvi dar início a este projeto de software para esta área.

A escolha do desenvolvimento de um software deste perfil foi baseada em uma intensa pesquisa na Internet sobre softwares na mesma linha de ação. Infelizmente foram encontrados poucos sistemas desenvolvidos para pequenos e médios investidores ou até mesmo que atendessem aos nossos padrões financeiros. Além disso, os diversos softwares financeiros encontrados no mercado são em maioria sistema desenvolvidos para empresas de grande porte, deixando de lado o usuário comum, ou até mesmo sistema que se encaixam em sistemas contábeis para empresas, ou seja, sempre pessoa jurídica.

1.3. Os objetivos

1.3.1. Objetivos preliminares

Os objetivos preliminares constituem-se basicamente do desenvolvimento de um sistema de computador utilizando as tecnologias que serão descritas no item 1.5. Este sistema terá como requisito funcional primordial o gerenciamento financeiro. Para isso o sistema será funcional, modular, usável por qualquer pessoa, confiável e seguro. O requisito modular é uma das premissas do desenvolvimento devido ao fato do sistema ter um cronograma de atualizações e refinamento dos algoritmos financeiros utilizados por ele. Esta modularidade pode ser vista na figura abaixo.

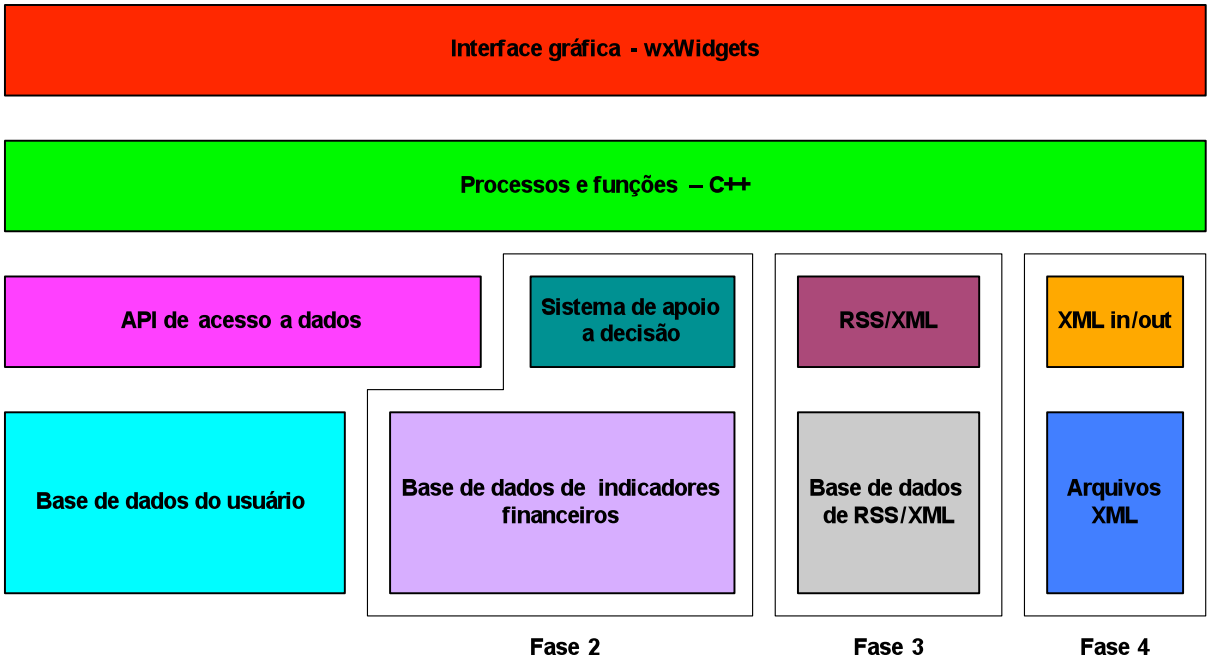

Figura 1 - Diagrama abstrato do sistema

A este diagrama foi dado o nome de abstrato devido á não retratar exatamente a arquitetura do sistema, um diagrama de arquitetura será discutido mais a frente, onde apresentarei os padrões e metodologias utilizadas no projeto.

No diagrama acima temos a camada inicial que é a interface gráfica do sistema, abaixo desta temos a camada que engloba os processos e funções do sistema. No terceiro nível do sistema temos a camada de acesso a dados e 3 componentes, a saber, o Sistema de apoio a decisão, que será o responsável pelos algoritmos financeiros, o *RSS/XML*, que terá como função apresentar e gerenciar aos *feeds* de notícias do sistema e por último um componente de entrada e saída de arquivos no padrão *XML*. Já a última camada compreende os dados utilizados pelo sistema bem como os arquivos *XML* de entrada e saída.

Note que o sistema terá 4 fases durante sua implementação, esta divisão foi criada para fins de controle de projeto e independência de componentes (modular). A fase 1 será o sistema em si com a possibilidade do usuário cadastrar suas contas, a fase 2 será a fase da implementação dos algoritmos financeiros. Na terceira fase temos a parte de notícias e por fim a fase 4 que é responsável pela entrada e saída de dados via arquivo *XML*.

Os algoritmos utilizados pelo sistema terão como base a análise de 4 índices, cambio (dólar), taxa de juros (*SELIC*), inflação (*IPCA* e *IGP-M*) e IBOVESPA. Todos os dados utilizados para a confecção de tais algoritmos virão de base de dados oficiais do governo, ou na sua falta, de instituições que tem como premissa fazer o levantamento de tais índices.

1.3.2. Objetivos futuros

Os objetivos futuros deste projeto consistem em melhorias incrementais de suas funcionalidades, bem como refinamento dos algoritmos financeiro a serem utilizados. Como o sistema será todo desenvolvido sob a ótima da modularidade e reuso a troca ou refinamento de um componente do sistema deve, a priori, não impactar no funcionamento do mesmo.

Coloquei como objetivo futuro o refinamento dos algoritmos financeiros devido a estes serem de extrema complexidade e não existirem padrões adotados largamente pelo mercado financeiro. A priori tais algoritmos serão criados através de regressão linear utilizando-se uma base de dados para cada área á ser contemplada pelo sistema. Futuramente realizarei maiores estudos sobre estes índices, sua composição, seus inter-relacionamentos, talvez como objeto de um mestrado.

1.4. Escopo do projeto

Como dito anteriormente o projeto foi divido em 4 fases que serão explicadas nos próximos itens.

1.4.1. Fase 1

A fase 1 do projeto consiste no desenvolvimento da interface gráfica, que será provida pela biblioteca wxWidgets que já possui inúmeras classes e métodos, além de ferramentas para construção de interfaces dentro dos padrões Windows ou Linux. Este biblioteca além de fornecer recursos para o ambiente gráfico possui também classes e métodos para acesso a dados, requisitos de configuração, plotagem de gráficos, etc.

Para implementação da API de acesso aos dados será utilizado o padrão DAO, descrito de melhor forma no item 1.5.2. Para o sistema de banco de dados será utilizada a biblioteca SQLite que é uma implementação de um banco de dados via linguagem C ANSI. Como esta biblioteca não é propriamente um banco de dados e por questões de em um futuro poder ser utilizado um SGBD, criarei uma camada intermediária entre os dados (SQLite) e o sistema. Esta interface é que será implementada utilizando o padrão de acesso a dados DAO.

1.4.2. Fase 2

A fase 2 será o coração do sistema, ou seja, a parte de análise financeira. Como dito anteriormente os algoritmos desta fase serão inicialmente rudimentares devido ao não conhecimento com maior profundidade de tais algoritmos. No mercado atualmente existem duas escola de análise financeira, a fundamentalista e a técnica. A primeira é largamente utilizada pelos analistas por ser um método mais apurado, contudo com maior complexidade devido à necessidade de análise de diversos artefatos contábeis, financeiros, índices de produção, custos, situação macroeconômica, etc.

Já o segundo método, o técnico ou fundamentalista, é de domínio mais simplista, visto que somente leva em conta o padrão de comportamento do mercado analisando seus

históricos e suas transações passadas. Este método pressupõem que todas as variáveis que podem afetar um índice ou ativo já estão incorporadas no seu preço, sendo assim o analista deve se ater ao grafismo do mercado financeiro para tentar determinar qual é o melhor momento de sair ou entrar no mercado. Este será o método utilizado pelo *PRP*.

1.4.3. Fase 3

A fase 3 do *PRP* consiste no desenvolvimento do *RSS/XML* que será utilizado para envio de mensagens e alertas aos usuários do sistema, por parte de um servidor de *RSS*. Este tipo de envio de informação foi escolhido devido a sua grande difusão na Internet, facilidade de utilização e atualização, bem como ser altamente customizado por parte do cliente.

O usuário poderá, depois de cadastrado no sistema, efetuar as configurações de quais alertas deseja receber e com que freqüência. Este alerta será visualizado pelo usuário através de um mini browser dentro de interface gráfica do *PRP*, que também poderá futuramente ser utilizado como sistema dinâmico de informações em tempo real.

1.4.4. Fase 4

Este fase tem a finalidade de criar um modulo que terá a função de gravar e ler arquivos *XML* para o sistema. A importação/exportação de arquivos em formato *XML* hoje em dia é de extrema importância devido a sua natureza de transferência de informação entre sistema que podem não ser compatíveis em um primeiro momento. Isso foi um dos motivos que me levou a incluir tal funcionalidade no *PRP*.

A gravação de arquivos *XML* será útil inicialmente para armazenar configurações do sistema e exportar relatórios. Já a leitura de arquivos *XML* será utilizada com a intenção de podermos entrar com informações neste formato no sistema, por exemplo, com um extrato/saldo bancário que possa ser gravado neste formato poderia ser importado para o *PRP* a fim de atualização de saldo.

1.5. Tecnologias utilizadas

1.5.1. Na análise

Como já mencionado anteriormente a modelagem do sistema foi feita utilizando-se a linguagem de modelagem unificada, *UML* (*Unified Modeling Language*), em sua versão 2.0. Esta tecnologia foi utilizada no projeto devido a ser uma linguagem desenvolvida para modelar sistema baseados no paradigma da analise e programação orientada a objetos, exatamente conceito utilizado no sistema *PRP*. Além disso, a *UML* proporciona uma analise e modelagem robusta, completamente desacoplada da técnica de implementação e tecnologia utilizada, preceitos que procurei seguir a fim de poder segmentar o projeto.

Na evolução do projeto utilizei o modelo evolucionário de processo de software, conhecido como modelo em espiral. Este modelo, proposto por Boehm em 1988, tem como conceito chave o desenvolvimento de projetos de software em ciclos (espirais) incrementais e com prototipagem ao final de cada ciclo. Estes ciclos de vida de projeto têm a finalidade de serem complementares uns aos outros, dando a possibilidade de se rever uma decisão tomada anteriormente afim de melhor adequá-la ao novo ciclo. Esta técnica de projeto tem uma vantagem inerente a sua concepção, a prototipagem, que vem a ser o produto de um ciclo completo o que dá uma melhor visão do andamento do projeto e já de início disponibiliza uma versão rudimentar do sistema final.

1.5.2. No projeto

Para a fase de projeto será utilizada a técnica de componentização utilizando *UML*. Esta moderna técnica vem sendo largamente utilizada em diversos projetos de software, com grande sucesso, devido a sua abordagem de componentes, ou seja, embasada em um dos principais pilares da engenharia de software orientada a objetos, baixo acoplamento e alta coesão. Este baixo acoplamento deve-se ao fato do componente ter seu funcionamento o mais independente dos outros componentes no sistema, somente utilizando um relacionamento entre tais quando necessário e por boa prática um relacionamento unidirecional. Já a alta coesão está pautada no conceito de responsabilidade da orientação a objetos, ou seja, um

objeto deve conter somente informações (atributos) e comportamentos (métodos) referentes ao seu assunto.

Esta metodologia de projeto foi extraída do livro *UML Components - A simple process for specifying component-based software de John Cheesman & John Daniels*. A componentização utilizando *UML* é chamada de Processo de Especificação de Componentes e é basicamente dividida em 3 fases de projeto, a saber, Identificação de Componentes, Interação de Componentes e Especificação de Componentes. Abaixo temos os gráficos das fases com os seus respectivos insumos e produtos.

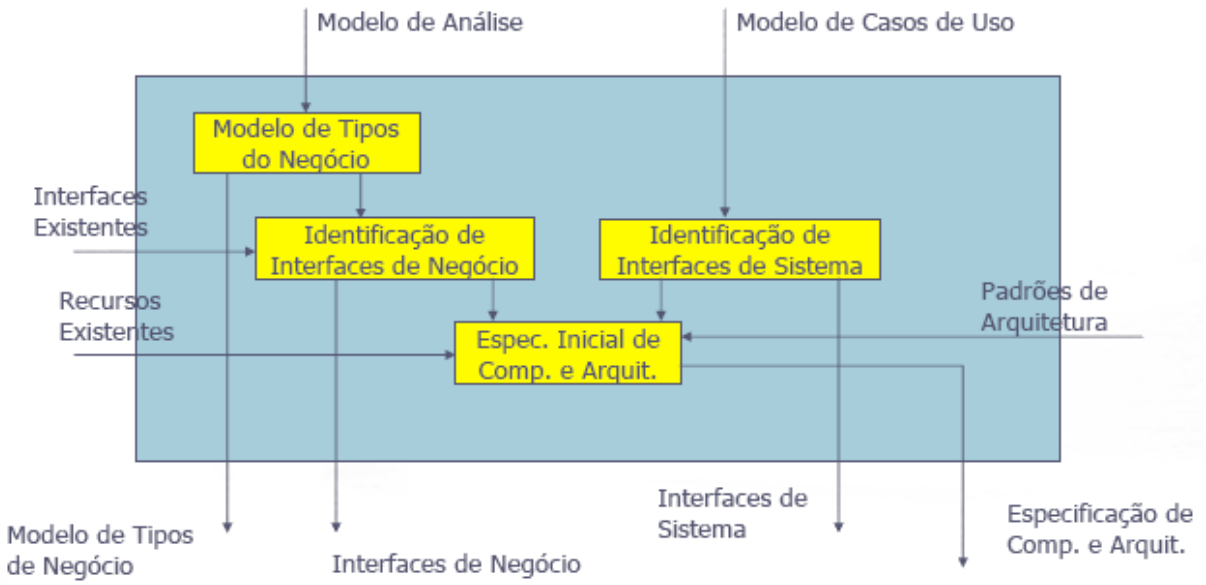

Figura 2 - Identificação de Componentes - Fase 1

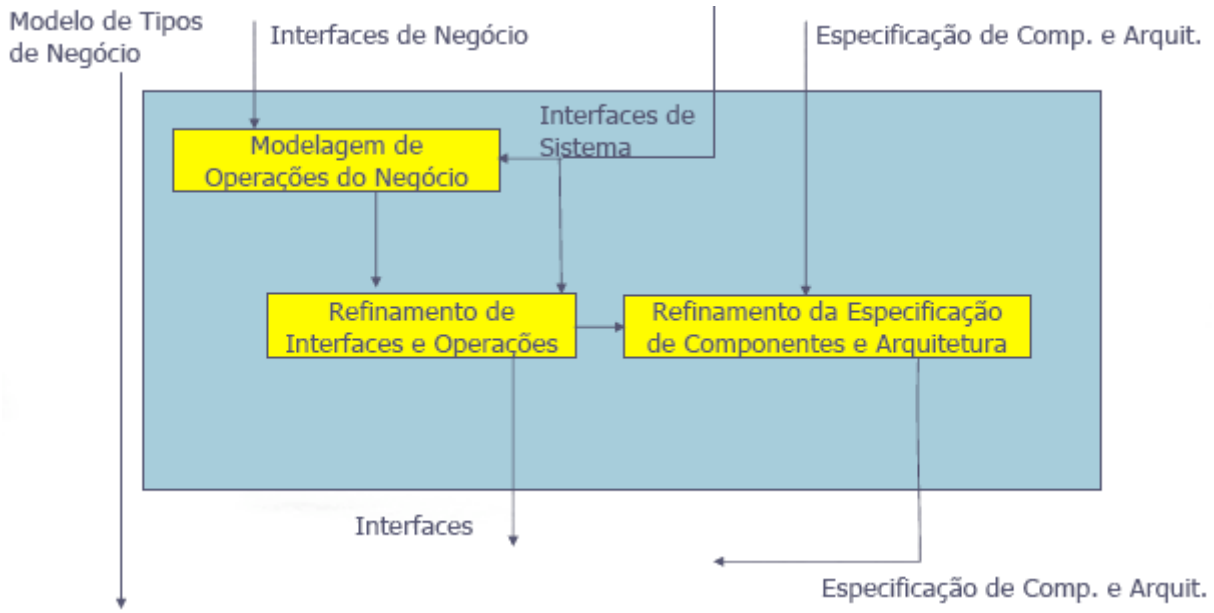

Figura 3 - Interação de Componentes - Fase 2

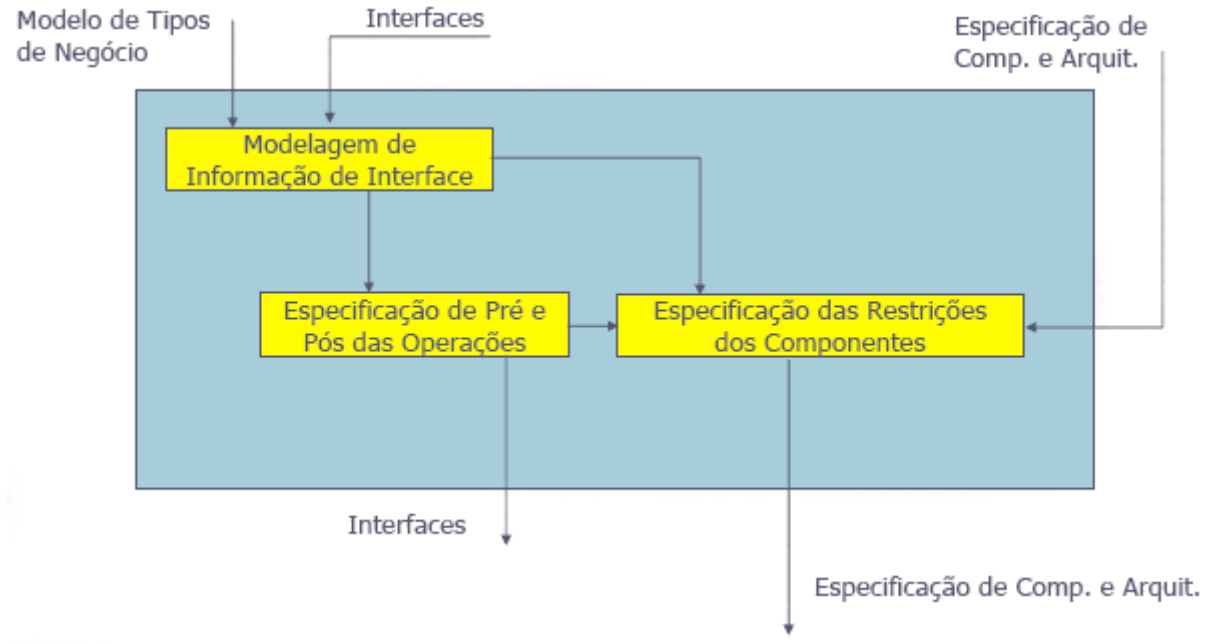

Figura 4 - Especificação de Componentes - Fase 3

A primeira fase, Identificação de Componentes, tem como objetivo identificar um conjunto de interfaces de negócios para os componentes de negócio e interface de sistemas para os componentes de sistema e montar uma arquitetura inicial. Esta fase tende a ser a de maior duração e esforço técnico, devido à necessidade de bons conhecimentos do negócio, de seu comportamento e principalmente de padrões de projetos e modelagem/análise *OO*.

A próxima fase, Interação de Componentes, tem como objetivo definir como cada uma das operações do sistema será realizada usando a arquitetura de componentes. Durante a evolução desta fase temos o diagrama de interação de componentes que é utilizado para a descoberta das operações necessárias para atender ás interfaces de negócio. Todas as responsabilidades são analisadas e colocadas nas interfaces corretas, minimizando assim as dependências e praticando a alta coesão do componente.

Por fim a última fase de projeto, Especificação de Componentes, irá detalhar as operações descobertas na fase anterior e determinará as restrições de uso dos componentes. Estas restrições são o detalhamento de como se deve executar uma operação provida por um componente, determinando qual estado/entrada deve ser fornecido a operação para que ela forneça um produto específico. Esta fase deve ser executada somente quando todas as operações e interfaces estiverem identificadas e a arquitetura for estável.

Todas as fases do processo de projeto acima descrito serão mais bem detalhadas e explicadas quando descrevermos o projeto do sistema *PRP* (vide item 3. Projeto).

1.5.3. Na implementação

Na implementação será utilizada o *framework* gráfico *wxWidgets*. É um *framework* composto por diversos *widgets*, que seriam os componentes ou elementos gráficos de uma interface com o usuário, tais como botões, janelas, barras de rolagem, menus, etc. Como o próprio autor e criador comente em seu livro, "*wxWidgets is a programmer's toolkit for writing desktop or mobile applications with graphical user interface (GUI). It's a framework, in the sense that it does a lot of the housekeeping work and provides default application behavior." – Cross-Plataform GUI Programming with wxWidgets, Juliam Smart and Kevin Hock with Stefan Csomor, Prentice Hall 2005*.

Este *framework* foi escolhido devido ao fato de já ter tido certo contato com seu estilo de programação durante a disciplina de Interface e Integração de Sistemas do Prof. Sérgio Barbosa Villas-Boas no Departamento de Engenharia Eletrônica da UFRJ. O *wxWidgets* foi lançado em 1992 por Julian Smart, que continua como seu principal desenvolvedor até os dias de hoje. O *framework* foi desenvolvido de forma a permitir que um programa seja compilado

e executado em diversas plataformas de computação, com nenhuma ou pouquíssimas modificações, ou seja, aplicações multi-plataforma.

O *wxWidgets* é implementado em C++, mas outras implementações estão disponíveis para várias das linguagens de programação mais comuns, entre elas, Python, Smalltalk, Perl e Java. wxWidgets é melhor descrita como um utilitário nativo. Ao invés de emular a apresentação de widgets utilizando primitivas gráficas nas diferentes plataformas suportadas, ela fornece uma pequena abstração para código nativo. Isso a torna mais rápida, e com um visual melhor adaptado à plataforma, que utilitários como o Java Swing.

Abaixo podemos ver a estrutura em camadas no qual se baseia o *framework wxWidgets*, note que, como dito acima, o programador faz chamadas a *API* do *wxWidgets* e não a *API* do sistema operacional. A *API* do *wxWidgets* trata as chamadas, interpretando-as e modificando-as, repassando tais chamadas a *API* do sistema operacional e recebendo deste as possíveis respostas.

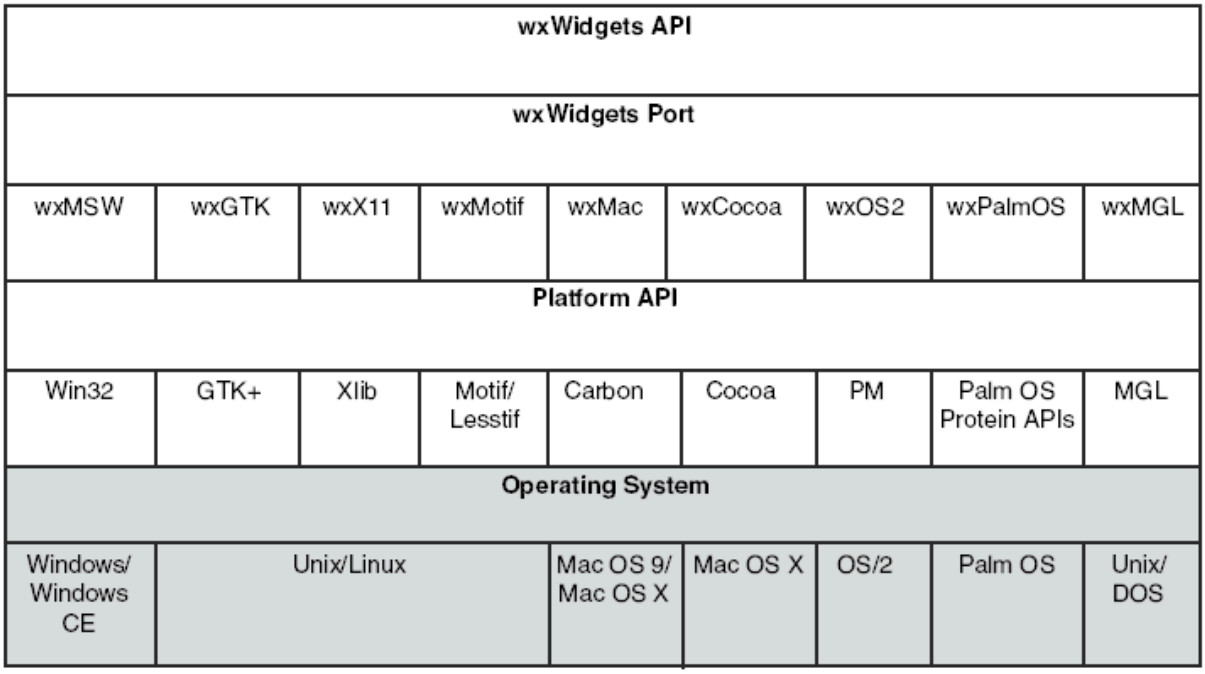

Figura 5 - Arquitetura do wxWidgets

Vemos no gráfico a riqueza de possibilidade que o *framework wxWidgets* permite ao desenvolvedor, podendo trabalhar com sistemas Windows, Unix/Linux, Mac OS e até mesmo sistemas embarcados, tais como Windows CE e Palm OS.

2. Análise

2.1. Levantamento de requisitos

O *PRP* propõe a ser um sistema de controle financeiro, com enfoque em controle de investimentos, sendo assim seus requisitos foram desenhados para atender este nicho de mercado. Muitos dos requisitos inicialmente levantados foram postergados devido à falta de tempo e conhecimento mais aprofundado no assunto.

O requisito principal do sistema é poder armazenar informações financeiras do usuário e disponibilizá-las a todo solicitação do usuário, sendo assim isso faz do *PRP* um sistema de informação. Além deste requisito o sistema também fará uma projeção financeira baseada em índices do mercado financeiro. Tais projeções serão baseadas no seguintes índices, *IGPM*, *IPCA*, *SELIC*, Bovespa e Câmbio. Esta funcionalidade que será desenvolvida para atender ao requisito também faz do *PRP* um sistema especialista, onde tem a finalidade de atuar como um "consultor" financeiro para o seu usuário.

O *PRP* deve ser amigável e utilizável por qualquer pessoa que tenha familiaridade com sistemas operacionais com ambiente gráfico. Deverá ser confiável, ou seja, somente permitir acesso as informações a quem de direito e ter disponibilidade, fornecendo as informações armazenadas na hora da solicitação do usuário. As contas deverão ser armazenadas de forma separada possibilitando o cadastro de lançamentos especificamente um uma conta. Todas as contas criadas no sistema deverão ser únicas em sua categoria, deverão ter nome e observação, além de data de criação.

Os lançamentos feitos nas contas deverão ter valor, serem identificados como de crédito ou débito, terem descrição e estarem associados somente a uma conta. Quando do apagamento de uma conta no sistema os seus respectivos lançamentos deverão ser igualmente apagados. O sistema deverá permitir o cadastro de usuários, estes usuários cadastrados deverão ser únicos, ou seja, não é permitido existir dois usuários com mesmo identificador no sistema.

2.2. Descrição das funcionalidades

As funcionalidades do *PRP* foram agrupadas em 5 pacotes, a saber; Gerenciamento de usuário, Gerenciamento de conta, Gerenciamento de lançamento, Gerenciamento financeiro e Gerenciamento de notícias.

O pacote de Gerenciamento de usuário será responsável pelas funções referentes a usuário, ou seja, cadastro, alteração, remoção e *login* no sistema. Já o pacote de Gerenciamento de conta terá as funções de cadastro, alteração, consulta e remoção de contas no sistema. O pacote de Gerenciamento de lançamento terá as mesmas funções do pacote de conta, aplicadas em cima do lançamento, e mais a funcionalidade de quitação de lançamento. No pacote de Gerenciamento de notícias temos o controle de configurações e leitura das mensagens em formato RSS.

O pacote de Gerenciamento financeiro será o pacote de maior importância no sistema devido a ser o que define a funcionalidade principal do sistema, o controle financeiro. Este pacote terá as funcionalidades básicas de cadastro, alteração, remoção e consulta das contas investimentos e as outras funcionalidades relacionadas à parte financeira do sistema, tais com a análise financeira, atualização de saldo e sincronismo de base de dados financeira.

2.3. Modelagem dos casos de uso

No processo de modelagem de casos de uso foi utilizada a linguagem de modelagem unificada (*UML*) com todas as suas peculiaridades e restrições. Durante o levantamento dos requisitos foram enumerados 23 casos de uso, como mencionados na tabela abaixo, e organizados em pacote como anteriormente descritos.

ID do caso de uso	Nome do caso de uso	Nome do pacote
001	EFETUAR LOGIN	Gerenciamento de usuário
002	CADASTRAR USUÁRIO	Gerenciamento de usuário
003	ALTERAR USUÁRIO	Gerenciamento de usuário
004	REMOVER USUÁRIO	Gerenciamento de usuário
005	CADASTRAR CONTA	Gerenciamento de conta
006	ALTERAR CONTA	Gerenciamento de conta
007	REMOVER CONTA	Gerenciamento de conta
008	CONSULTAR CONTA	Gerenciamento de conta
009	CADASTRAR LANÇAMENTO	Gerenciamento de lançamento
010	ALTERAR LANÇAMENTO	Gerenciamento de lançamento
011	REMOVER LANÇAMENTO	Gerenciamento de lançamento
012	CONSULTAR LANÇAMENTO	Gerenciamento de lançamento
013	QUITAR LANÇAMENTO	Gerenciamento de lançamento
014	CADASTRAR CONTA INVESTIMENTO	Gerenciamento financeiro
015	ALTERAR CONTA INVESTIMENTO	Gerenciamento financeiro
016	REMOVER CONTA INVESTIMENTO	Gerenciamento financeiro
017	CONSULTAR CONTA INVESTIMENTO	Gerenciamento financeiro
018	SOLICITAR ANÁLISE FINANCEIRA	Gerenciamento financeiro
019	ATUALIZAR BASE DE DADOS	Gerenciamento financeiro
020	ATUALIZAR SALDO	Gerenciamento financeiro
021	CONFIGURAR RSS	Gerenciamento de notícias
022	ATUALIZAR RSS	Gerenciamento de notícias
023	LER NOTÍCIA RSS	Gerenciamento de notícias

Tabela 1 - Casos de uso e seus pacotes

Abaixo temos o diagrama de casos de uso do sistema PRP.

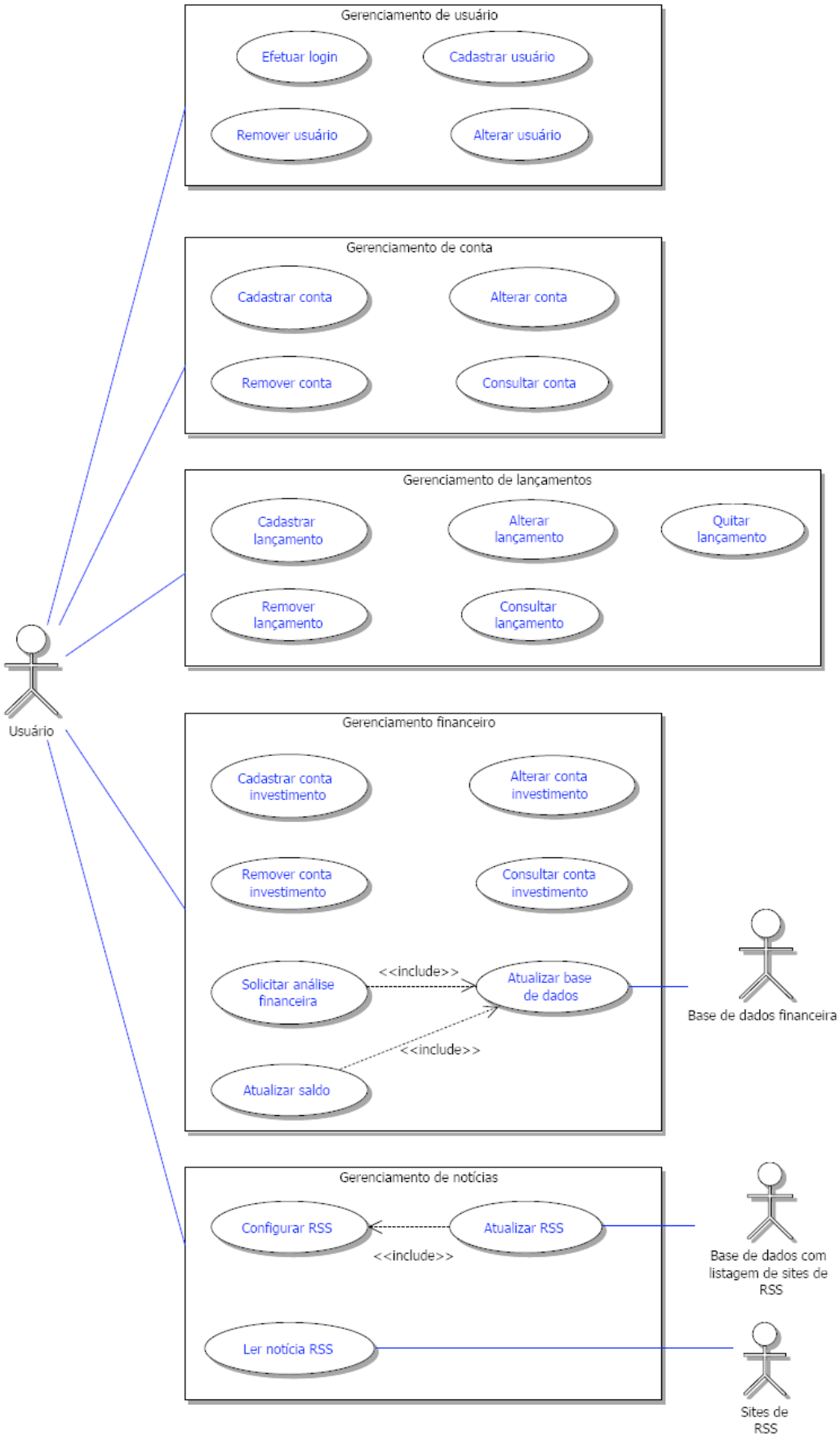

Figura 6 - Diagrama de casos de uso do PRP

2.4. Descrição dos casos de uso

Informações	
ID do caso de uso:	001
Nome do caso de uso:	EFETUAR LOGIN
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário de conectar-se ao sistema. Ao ser iniciado o sistema solicita ao usuário seu login (usuário e senha) através de uma tela de login. Esta tela de login é composta por dois campos a serem preenchidos, usuário e senha e dois botões denominados Ok e Cancel. Na situação do usuário ser validado no processo de login o sistema libera o acesso aos seus dados pessoais. Na situação do usuário não ser validado no processo de login o sistema entra em funcionamento normal, mas sem nenhum usuário validado. Note que neste ponto o usuário nada pode fazer no sistema a não ser cadastrar um novo usuário.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
	1 – Disponibilizar tela de login;
2 – Digitar usuário e senha;	
	3 – Liberar acesso ao sistema com o usuário validado;
	4 – Informar ao usuário acesso validado;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	3 – Informar ao usuário acesso inválido;
	4 – Liberar acesso ao sistema sem o usuário validado;

Informações	
ID do caso de uso:	002
Nome do caso de uso:	CADASTRAR USUÁRIO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário de cadastrar-se no sistema. Quando o sistema á iniciado pela primeira vez automaticamente ele entra no modo cadastro de usuário. Esta condição é criada devido ao fato de para operar o sistema ser necessário pelo menos um usuário cadastrado. O sistema poderá aceitar diversos usuários cadastrados tendo somente como limitador a quantidade de espaço em disco rígido onde está instalado. O processo de cadastro, tanto do primeiro usuário, quanto dos demais, será feito através de assistente de cadastro. Este assistente consistirá em um conjunto de telas encadeadas que irão solicitando ao usuário informações necessárias para realizar o seu cadastro. Estas informações são: - nome de usuário (20 caracteres alfabéticos excluindo caracteres acentuados); - senha (8 caracteres alfanuméricos excluindo caracteres acentuados); - nome completo (255 caracteres alfabéticos incluindo caracteres acentuados); - e-mail (20 caracteres padrão RFC 2821/2822); - localização geográfica (listagem de estados nacionais); - idioma de utilização do sistema (listagem de idiomas disponíveis); - numero do banco (4 caracteres – seleção de listagem); Durante o processo de cadastro o usuário poderá navegar pelas telas do assistente através dos botões BACK e NEXT, e também cancelar o processo clicando no botão CANCEL, ou teclando ESC. A tecla ENTER estará sempre associada ao botão NEXT e na tela final ao botão FINISH. Ao término do processo de cadastro será mostrada uma tela com os dados digitados pelo usuário, para conferencia, e após a confirmação o usuário deverá clicar no botão FINISH o que encerra o cadastro.
Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**

	1 – Exibir a tela inicial do assistente de cadastro;
2 – Aceita as condições de cadastro;	
	3 – Exibir a primeira tela de cadastro;
4 – Preencher os campos com as informações solicitadas;	
	5 – Exibir a segunda tela de cadastro;
6 – Preencher os campos com as informações solicitadas;	
	7 – Exibir a tela de confirmação das informações;
8 – Confirma o cadastro;	
	9 – Informa cadastro bem sucedido;
Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	9 – Informa usuário já existente;

Informações	
ID do caso de uso:	003
Nome do caso de uso:	ALTERAR USUÁRIO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário alterar suas informações cadastrais. As informações que o usuário poderá alterar são; nome completo, e-mail, localização geográfica, idioma de utilização do sistema e as informações bancárias. Senha e nome de usuário não poderão ser alterados neste processo. Como para alterar as informações de cadastro de um usuário é necessário que o mesmo esteja autenticado no sistema, sendo assim cada usuário somente poderá alterar suas informações cadastrais.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar alteração das informações cadastrais de um usuário;	
	2 – Exibir dados do usuário atualmente cadastrado;
3 – Efetuar as modificações;	
	4 – Informar que as alterações foram efetuadas;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	4 – Informar que as alterações não foram efetuadas;

Informações	
ID do caso de uso:	004
Nome do caso de uso:	REMOVER USUÁRIO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário remover suas informações do sistema. Por questões de segurança será necessário digitar a senha de cadastro para efetuar a remoção dos dados cadastrais bem como todas as informações sobre o usuário.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar remoção de usuário;	
	2 – Listar todos os usuários cadastrados;
3 – Selecionar o usuário a ser removido;	
	4 – Solicitar senha do usuário selecionado;
5 – Digitar senha do usuário a ser removido;	
	6 – Solicitar confirmação e avisar que serão removidas todas as contas e os lançamentos do usuário;
7 – Confirmar;	
	8 – Informar que a remoção foi efetuada com sucesso;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	6 – Informar senha inválida;

Fluxo Alternativo de Eventos 2	
Ações do Ator:	**Ações do Sistema:**
	8 – Informar que a remoção não foi efetuada;

Fluxo Alternativo de Eventos 3	
Ações do Ator:	**Ações do Sistema:**
7 – Cancelar operação;	
	8 – Informar que nenhum usuário removido;

Informações	
ID do caso de uso:	005
Nome do caso de uso:	CADASTRAR CONTA
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário de cadastrar uma conta no sistema. Uma conta é uma composição de lançamentos podendo ser de 2 naturezas, débito ou crédito. Durante o processo de cadastro o usuário irá inserir informações referentes à conta. Estas informações são: - natureza da conta (conta de despesa ou conta de receita); - nome da conta (seleção dentre opções com possibilidade de adição de título); - data da conta (data 8 dígitos – data da sua criação); - observação (255 caracteres ASCII). Obs.: título da conta: luz, gás, telefone, aluguel, etc.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar cadastro de conta;	
	2 – Informar naturezas de conta;
3 – Selecionar natureza da conta;	
	4 – Solicitar dados da conta;
5 – Informar dados da conta;	
	6 – Informar conta cadastrada com sucesso;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	6 – Informar conta não cadastrada;

Informações	
ID do caso de uso:	006
Nome do caso de uso:	ALTERAR CONTA
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário de alterar as informações de uma conta. Quando uma conta cadastrada já contiver lançamentos somente será permitido alterar seu campo observação. Caso a conta ainda não tenha lançamento feitos todos os seus dados podem ser alterados.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar alteração de conta;	
	2 – Listar contas cadastradas;
3 – Selecionar conta a ser alterada;	
	4 – Exibir dados da conta com opção de alterar os campos permitidos;
5 – Efetuar as alterações;	
6 – Solicitar que as alterações sejam salvas;	
	7 – Informar que as alterações foram feitas com êxito;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	7 – Informar que as alterações não foram feitas;

Informações	
ID do caso de uso:	007
Nome do caso de uso:	REMOVER CONTA
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se ao ato do usuário remover uma conta do sistema. Por questões de segurança será necessário que o usuário digite sua senha para confirmar a remoção de uma conta do sistema. Quando uma conta for removida todos os seus lançamentos serão removidos também.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar remoção de conta;	
	2 – Listar todas as contas cadastradas;
3 – Selecionar conta a ser removida;	
	4 – Solicitar senha do usuário;
5 – Digitar senha do usuário;	
	6 – Solicitar confirmação e avisar que serão removidos todos os lançamentos referentes à conta a se removida;
7 – Confirmar;	
	8 – Informar que a remoção foi efetuada com sucesso;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	6 – Informar senha inválida;

Fluxo Alternativo de Eventos 2	
Ações do Ator:	**Ações do Sistema:**
	8 – Informar que a remoção não foi efetuada;

Fluxo Alternativo de Eventos 3	
Ações do Ator:	**Ações do Sistema:**
7 – Cancelar operação;	
	8 – Informar que nenhuma conta removida;

Informações	
ID do caso de uso:	008
Nome do caso de uso:	CONSULTAR CONTA
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário de consultar uma conta cadastrada no sistema. As consultas no sistema serão parametrizadas através dos seguintes campos: - nome da conta; - natureza da conta; - data de criação (podendo ser por dia exato, mês ou ano); Ao final da consulta será apresentada ao usuário uma listagem das contas encontradas com opção de visualizar seus lançamentos.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar consulta de conta;	
	2 – Apresentar tipos de consultas;
3 – Selecionar tipo da consulta;	
	4 – Solicitar dados da consulta;
5 – Informar dados da consulta;	
	6 – Apresentar resultados da consulta;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	6 – Informar erro de consulta;

Informações	
ID do caso de uso:	009
Nome do caso de uso:	CADASTRAR LANÇAMENTO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário em cadastrar um lançamento em uma conta de despesa ou receita. Durante o processo de cadastro do lançamento serão solicitados os seguintes dados: - conta do lançamento; - tipo do lançamento (débito/crédito); - valor (moeda corrente); - data da liquidação (data); - descrição (255 caracteres ASCII); Caso um lançamento seja incluído no sistema com a data atual, este somente poderá ser editado até o final do dia. Caso o usuário já cadastre o lançamento liquidado este se tornará inalterável automaticamente.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar cadastro de lançamento;	
	2 – Listar as contas para lançamento;
3 – Selecionar a conta a qual irá inserir o lançamento;	
	4 – Solicitar dados do lançamento;
5 – Informar dados do lançamento;	
	6 – Informar lançamento efetuado com êxito;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	6 – Informar lançamento não efetuado;

Informações	
ID do caso de uso:	010
Nome do caso de uso:	ALTERAR LANÇAMENTO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário de alterar um lançamento ainda em aberto. Um lançamento em aberto significa um lançamento em que a data de liquidação ainda não chegou, ou seja, o lançamento não foi liquidado. Todos os dados de um lançamento não liquidado podem ser alterados.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar alteração de lançamento;	
	2 – Listar contas cadastradas;
3 – Selecionar a conta ao qual será alterado o lançamento;	
	4 – Exibir dados do lançamento com opção de alterar os campos permitidos;
5 – Efetuar as alterações;	
6 – Solicitar que as alterações sejam salvas;	
	7 – Informar que as alterações foram feitas com êxito;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	7 – Informar que as alterações não foram feitas;

Informações	
ID do caso de uso:	011
Nome do caso de uso:	REMOVER LANÇAMENTO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se ao ato do usuário remover um ou mais lançamentos de uma conta. Um lançamento somente poderá ser removido se ainda não foi liquidado.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar remoção de lançamento;	
	2 – Listar contas cadastradas;
3 – Selecionar conta a qual deseja remover o lançamento;	
	4 – Exibir todos os lançamentos;
5 – Selecionar o(s) lançamento(s) que se deseja remover;	
	6 – Solicitar confirmação de remoção de lançamento(s);
7 – Confirmar;	
	8 – Informar que a remoção foi feita com êxito;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	8 – Informar lançamento já quitado;

Fluxo Alternativo de Eventos 2	
	8 – Informar que a remoção não foi feita;

Fluxo Alternativo de Eventos 3	
Ações do Ator:	**Ações do Sistema:**
7 – Cancelar operação;	
	8 – Informar que nenhum lançamento removido;

Informações	
ID do caso de uso:	012
Nome do caso de uso:	CONSULTAR LANÇAMENTO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário de consultar um lançamento em uma determinada conta. As consultas no sistema serão parametrizadas através dos seguintes campos: - pertencentes a uma conta; - tipo do lançamento (débito/crédito); - valor (moeda corrente); - data da liquidação (data); - texto contido na descrição (255 caracteres ASCII); Ao final da consulta será apresentada ao usuário uma listagem dos lançamentos encontrados.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar consulta de lançamento;	
	2 – Apresentar tipos de consultas;
3 – Selecionar tipo da consulta;	
	4 – Solicitar dados da consulta;
5 – Informar dados da consulta;	
	6 – Apresentar resultados da consulta;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	6 – Informar erro na consulta;

Informações	
ID do caso de uso:	013
Nome do caso de uso:	QUITAR LANÇAMENTO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário quitar um lançamento. A quitação de um lançamento é feita solicitando ao sistema que informe os lançamentos em aberto (não quitados), podendo ou não utilizar-se de um filtro, e assinalando os lançamentos que se deseja quitar. Um lançamento depois de quitado não pode ser removido do sistema. A quitação somente pode ser feita através da data, ou seja, a data de quitação deve ser obrigatoriamente igual ou maior que a data prevista para a quitação. Os filtros de consulta serão: - por conta; - por data prevista; - por valor; - por tipo; - em aberto;

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicita listagem de lançamentos em aberto;	
	2 – Apresentar os possíveis filtros e consulta;
3 – Selecionar um filtro de consulta (opcional)	
	4 – Listar os resultados obtidos utilizando o filtro selecionado;
5 – Selecionar os lançamentos que deseja quitar;	
	6 – Solicitar confirmação de quitação de lançamentos;
7 - Confirmar	
	8 – Informar quitação efetuada com sucesso;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	8 – Informar erro na quitação;

Informações	
ID do caso de uso:	014
Nome do caso de uso:	CADASTRAR CONTA INVESTIMENTO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário de cadastrar uma conta investimento no sistema. Uma conta investimentos é uma composição de lançamentos de débitos e créditos. Durante o processo de cadastro o usuário irá inserir informações referentes à conta. Estas informações são: - tipo da conta investimento (seleção dentre opções); - nome da conta investimento (seleção dentre opções); - IOF (sim ou não); - IR (sim ou não); - CPMF (sim ou não); - índice de referencia (seleção dentre opções); - data da conta investimento (data 8 dígitos – data da sua criação); - observação (255 caracteres ASCII). Obs.: tipo da conta investimento: fundos, CDB, poupança, títulos públicos ou ações. Cada um dos tipos possíveis terá suas particularidades de informações cadastrais.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar cadastro de conta investimento;	
	2 – Informar possíveis tipos de conta investimento;
3 – Selecionar um tipo de conta investimento;	
	4 – Solicitar dados da conta investimento;
5 – Informar dados da conta investimento;	
	6 – Informar conta investimento cadastrada com sucesso;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	6 – Informar conta investimento não cadastrada;

Informações	
ID do caso de uso:	015
Nome do caso de uso:	ALTERAR CONTA INVESTIMENTO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário de alterar as informações de uma conta.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar alteração de conta investimento;	
	2 – Listar tipos de conta investimento cadastradas;
3 – Selecionar conta investimento a ser alterada;	
	4 – Exibir dados da conta investimento com opção de alterar os campos permitidos;
5 – Efetuar as alterações;	
6 – Solicitar que as alterações sejam salvas;	
	7 – Informar que as modificações foram feitas com êxito;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	7 – Informar que as modificações não foram feitas;

Informações	
ID do caso de uso:	016
Nome do caso de uso:	REMOVER CONTA INVESTIMENTO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se ao ato do usuário remover uma conta do sistema. Por questões de segurança será necessário que o usuário digite sua senha para confirmar a remoção de uma conta do sistema. Quando uma conta for removida todos os seus lançamentos serão removidos também.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar remoção de conta investimento;	
	2 – Listar todas as contas investimento cadastradas;
3 – Selecionar conta investimento a ser removida;	
	4 – Solicitar senha do usuário;
5 – Digitar senha do usuário;	
	6 – Solicitar confirmação e avisar que serão removidos todos os lançamentos referentes à conta investimento a se removida;
7 – Confirmar;	
	8 – Informar que a remoção foi efetuada com sucesso;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	6 – Informar senha inválida;

Fluxo Alternativo de Eventos 2	
Ações do Ator:	**Ações do Sistema:**
	8 – Informar que a remoção não foi efetuada;

Fluxo Alternativo de Eventos 3	
Ações do Ator:	**Ações do Sistema:**
7 – Cancelar operação;	
	8 – Informar que nenhuma conta investimento removida;

Informações	
ID do caso de uso:	017
Nome do caso de uso:	CONSULTAR CONTA INVESTIMENTO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário de consultar uma conta cadastrada no sistema. As consultas no sistema serão parametrizadas através dos seguintes campos: - nome da conta investimento; - tipo da conta investimento; - data de criação (podendo ser por dia exato, mês ou ano); Ao final da consulta será apresentada ao usuário uma listagem das contas encontradas com opção de visualizar seus lançamentos.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar consulta de conta investimento;	
	2 – Apresentar tipos de consultas;
3 – Selecionar tipo da consulta;	
	4 – Solicitar dados da consulta;
5 – Informar dados da consulta;	
	6 – Apresentar resultados da consulta;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	6 – Informar erro de consulta;

Informações	
ID do caso de uso:	018
Nome do caso de uso:	SOLICITAR ANÁLISE FINANCEIRA
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário em solicitar ao sistema que este faça uma estimativa de índices financeiros. Esta análise será feita em 4 áreas; cambio (dólar), taxa de juros (SELIC), inflação (IPCA e IGP-M) e IBOVESPA. As estimativas serão feitas através de algoritmos próprios que utilizam métodos de previsão de índices financeiros, com base em dados financeiros armazenados. As informações para a análise financeira estarão armazenadas localmente. Após a análise financeira o usuário poderá aplicá-la a uma conta investimento que desejar.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar análise financeira;	
	2 – Executar: ATUALIZAR BASE DE DADOS;
	3 – Listar os tipos de análise financeira;
4 – Selecionar qual índice financeiro deseja estimar;	
5 – Selecionar qual período para a estimação;	
	6 – Apresentar a estimativa do índice e seus erros (média e desvio padrão);

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	2 – Erro em ATUALIZAR BASE DE DADOS;

Fluxo Alternativo de Eventos 2	
Ações do Ator:	**Ações do Sistema:**
	6 – Informar que não foi possível realizar estimativa;

Informações	
ID do caso de uso:	019
Nome do caso de uso:	ATUALIZAR BASE DE DADOS
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	BASE DE DADOS FINANCEIROS
Descrição:	O processo refere-se à ação da atualizar a base de dados financeira local do sistema. Esta base de dados será a origem dos dados que serão utilizados no processo de estimativa de índices financeiros. Esta base local será atualizada através de uma base de dados financeira servidora que será mantida pelo desenvolvedor do sistema. As atualizações serão feitas em pacotes que serão disponibilizados em períodos ainda a serem determinados. Estas atualizações não serão acumulativas. Os dados a serem atualizados serão os índices financeiros cambio (dólar), taxa de juros (SELIC), inflação (IPCA e IGP-M) e IBOVESPA.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
	1 – Solicitar conexão com o servidor de banco de dados financeiros;
2 – Realizar conexão;	
3 – Solicita senha de acesso;	
	4 – Enviar senha de acesso;
5 – Liberar acesso;	
	6 – Enviar situação atual das atualizações locais;
7 – Enviar atualização;	
	8 – Atualizar base local com os novos dados;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
2 – Informar impossibilidade de realizar conexão;	
	3 – Sair do procedimento de atualização e informar;

Fluxo Alternativo de Eventos 2	
Ações do Ator:	**Ações do Sistema:**
5 – Acesso negado;	

	6 – Sair do procedimento de atualização e informar;
Fluxo Alternativo de Eventos 3	
Ações do Ator:	**Ações do Sistema:**
7 – Não há atualizações a serem feitas;	
	8 – Sair do procedimento de atualização e informar;

Informações	
ID do caso de uso:	020
Nome do caso de uso:	ATUALIZAR SALDO
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário em atualizar o saldo de uma conta investimento baseado nos índices financeiros armazenados na base de dados financeiros. Como cada conta tem atrelada a si um índice, fornecido na sua criação, é esta taxa que será utilizada para atualizar o saldo das contas investimentos. Como a atualização depende da base local estar sincronizada com a base remota, esta condição de sincronismo deve ser satisfeita antes da atualização do saldo.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar atualização de saldo das contas investimento;	
	2 – Executar: ATUALIZAR BASE DE DADOS;
	3 – Atualizar saldo das contas investimento;
	4 – Informar saldo atualizado;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**

Informações	
ID do caso de uso:	021
Nome do caso de uso:	CONFIGURAR RSS
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO
Descrição:	O processo refere-se à ação do usuário de configurar o sistema para recebimento de RSS no formato XML. Esta configuração está relacionada à seleção de sites de notícias que o sistema receberá informações. O sistema PRP terá o seu próprio site de notícias financeiras o qual já vem configurado na instalação. A configuração será feita selecionando-se os sites, que somente serão de notícias financeiras. O usuário ainda poderá determinar quantas mensagens devem ser armazenadas no painel de leitura. Este dois itens de configuração (seleção de sites e quantidade de noticias no painel) serão obrigatórios para a leitura de notícias.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicita configuração de RSS;	
	2 – Listar todos os sites de noticias cadastrados;
3 – Selecionar os sites os quais deseja receber notícias;	
4 – Determinar a quantidade de notícias a serem armazenadas no painel de leitura;	
	5 – Informar que a configuração dos sites selecionados foi executada com sucesso;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	5 – Informar erro na a configuração dos sites de RSS;

Informações	
ID do caso de uso:	022
Nome do caso de uso:	ATUALIZAR RSS
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO, BASE DE DADOS COM LISTAGEM DE SITES DE RSS
Descrição:	O processo refere-se à ação do usuário de atualizar a base de dados de site de notícias que o PRP pode acessar. Esta atualização é feita através de conexão TCP/IP com o servidor de RSS que será responsável por atualizar a listagem de sites de notícias financeiras os quais o PRP pode receber informações. O arquivo de atualização terá o formato XML.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
1 – Solicitar atualização de listagem de site RSS;	
	2 – Informar atualização executada com sucesso;
	3 – Executar: CONFIGURAR RSS;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	2 – Informar que não há atualização a ser feita;
	1

Informações	
ID do caso de uso:	023
Nome do caso de uso:	LER NOTÍCIA RSS
Criado por:	Marcelo V. V. Magalhães
Data da última atualização:	06/12/2009
Atores:	USUÁRIO, SITES DE RSS
Descrição:	O processo refere-se à ação do usuário de ler as notícias RSS no sistema. Depois de configurar o sistema de notícias o usuário poderá ler os *feeds* enviados pelos sites assinados. Esta leitura será feita em um painel na interface gráfica do PRP, o qual exibirá a quantidade de notícias que o usuário configurou.

Fluxo Básico de Eventos	
Ações do Ator:	**Ações do Sistema:**
	1 – Exibir o *feed* de notícia para leitura;

Fluxo Alternativo de Eventos 1	
Ações do Ator:	**Ações do Sistema:**
	1 – Informar que não é possível exibir o *feed* de notícia;

2.5. Modelagem das classes de domínio

Na página seguinte temos o modelo de classes de domínio. Neste modelo estão representados somente os atributos das classes e suas associações dentro do contexto do negócio. Mais adianta, na fase de projeto, serão especificados os métodos e suas assinaturas. Este diagrama de classes foi modelado sem enfocar o projeto e/ou implementação, exatamente como deve ser feito, visto que certas classes futuramente poderiam ou não ser incorporadas a outras se tornando atributos desta, ou serem transformadas em classes descritivas, por exemplo.

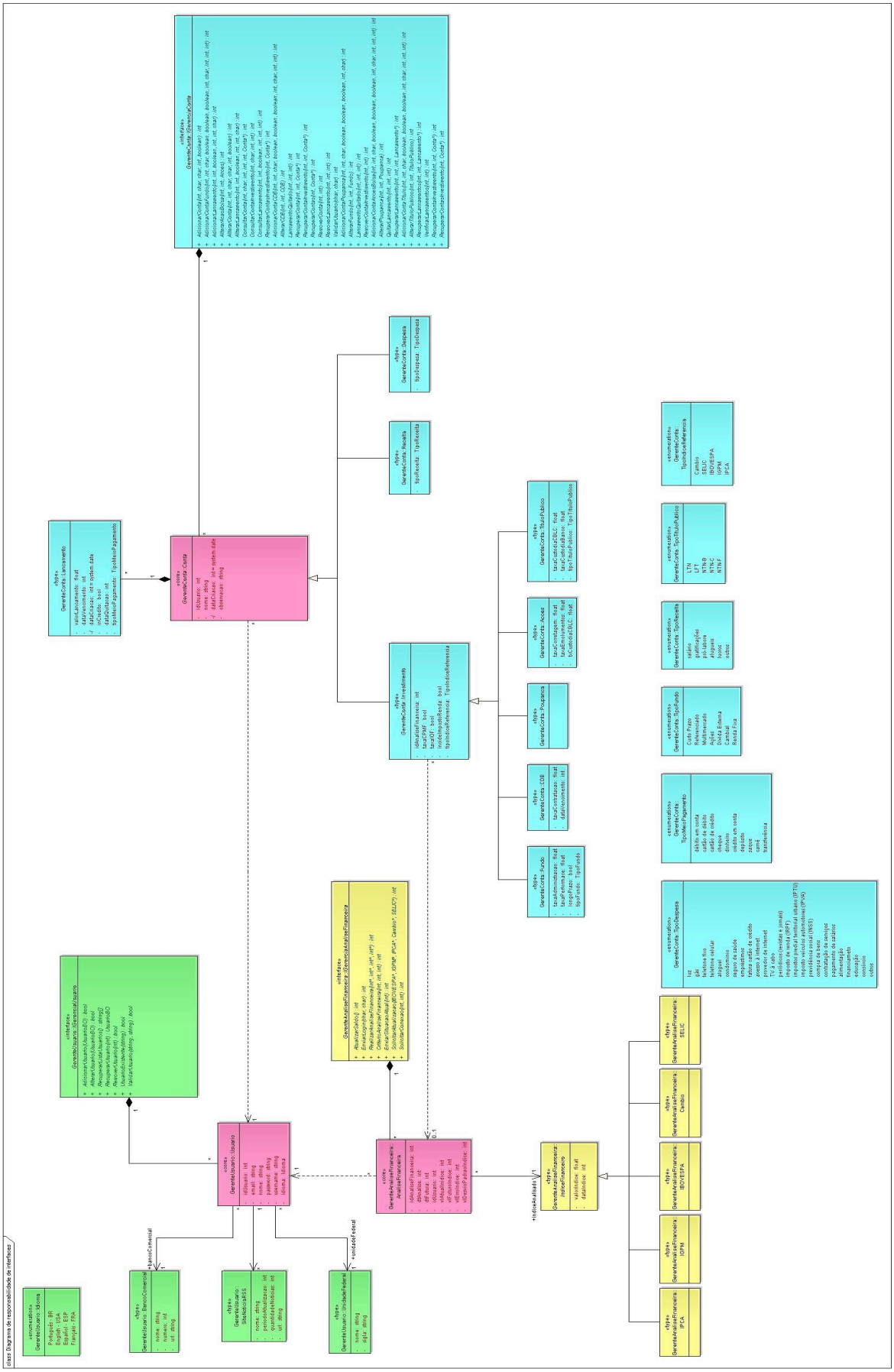

Figura 7 - Diagrama de classes de domínio

2.6. Descrição das classes de domínio e seus atributos

Abaixo temos o dicionário de dados do PRP com a descrição das classes e de seus atributos, incluindo o domínio, a visibilidade, o tamanho e a descrição.

Classe: Usuario

Descrição: esta classe será responsável pelos métodos e atributo relacionados aos usuários do sistema. Um usuário cadastrado no sistema pode criar contas e efetuar lançamentos em conta existentes, além disso, o usuário terá associado a ele um idioma de utilização do sistema, a sua localização (estado) e um banco comercial cadastrado.

Atributo: username
Domínio: string de caracteres alfanuméricos
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Nome do usuário no sistema. Este nome será composto somente por caracteres alfanuméricos minúsculos e terá a finalidade de identificar um usuário único no sistema.

Atributo: password
Domínio: string de caracteres alfanuméricos
Visibilidade: privada
Tamanho: 6 caracteres
Descrição: Senha do usuário no sistema. Esta senha deverá conter números e letras e será armazenada de forma criptografada.

Atributo: nome
Domínio: string de caracteres ASCII
Visibilidade: privada
Tamanho: 50 caracteres
Descrição: Nome completo do usuário. Este nome poderá ser diferente, ou não, do username, sendo utilizado para fins de cadastro no sistema.

Atributo: email
Domínio: string de caracteres no padrão RFC 2821/2822
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Endereço de e-mail do usuário.

Atributo: mostrarDica
Domínio: verdadeiro/falso
Visibilidade: privada
Tamanho: booleano
Descrição: Indicador se a dica do dia será mostrada ao usuário. Caso verdadeiro (1, true, sim) indica que será mostrada a dica do dia ao usuário assim que ele se conectar. Caso falso (0, false, não) indica que não será mostrada a dica do dia ao usuário.

Atributo: idioma
Domínio: enumeração - Idioma

Visibilidade: privada
Tamanho: 1dígito
Descrição: Identificador do idioma que o usuário quer que o sistema seja apresentado a ele.

Classe: BancoComercial

Descrição: este classe será responsável por gerenciar a listagem de bancos comerciais existentes. Um banco comercial pode estar associado a mais de um usuário.

Atributo: numero
Domínio: inteiro
Visibilidade: privada
Tamanho: 3 dígitos
Descrição: Código de um banco comercial no sistema nacional de compensação de títulos. Este código irá servir para identificar um banco comercial único no sistema.

Atributo: nome
Domínio: string de caracteres ASCII
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Nome comercial do banco que o usuário irá cadastrar no sistema.

Atributo: url
Domínio: string de caracteres no padrão RFC 1630
Visibilidade: privada
Tamanho: 50 caracteres
Descrição: Endereço web do site do banco comercial que o usuário irá cadastrar no sistema.

Classe: UnidadeFederal

Descrição: esta classe terá a função de gerenciar as possíveis unidades federais que um usuário pode estar localizado. Uma unidade federal (estado) pode estar associada a mais de um usuário.

Atributo: nome
Domínio: string de caracteres ASCII
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Nome da unidade federativa brasileira (estados).

Atributo: sigla
Domínio: string de caracteres alfabéticos (em maiúsculo)
Visibilidade: privada
Tamanho: 2 caracteres
Descrição: Sigla da unidade federativa brasileira (estados).

Classe: SiteNoticiaRSS

Descrição: esta classe terá a função de gerenciar as possíveis unidades federais que um usuário pode estar localizado. Uma unidade federal (estado) pode estar associada a mais de um usuário.

Atributo: nome
Domínio: string de caracteres ASCII
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Nome da unidade federativa brasileira (estados).

Atributo: periodoAtualizacao
Domínio: inteiro
Visibilidade: privada
Tamanho: 2 dígitos
Descrição: Período, em minutos, em que serão verificadas as novas atualizações no site de notícias.

Atributo: quantidadeNoticias
Domínio: inteiro
Visibilidade: privada
Tamanho: 2 dígitos
Descrição: Quantidade de noticias que serão apresentadas ao usuário.

Atributo: url
Domínio: string de caracteres no padrão RFC 1630
Visibilidade: privada
Tamanho: 50 caracteres
Descrição: Endereço web do site de notícias que o usuário irá cadastrar no sistema.

Classe: Conta

Descrição: esta classe é uma generalização dos possíveis tipos de contas que o sistema irá gerenciar, sendo assim uma classe abstrata. Uma conta pertence somente a um usuário e é composta por diversos lançamentos.

Atributo: nome
Domínio: string de caracteres ASCII
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Nome da conta.

Atributo: dataCriacao
Domínio: data
Visibilidade: privada
Tamanho: 6 dígitos
Descrição: Data da criação da conta (formato: dd/mm/aaaa). Este atributo é derivado, devido a ser a data atual no sistema.

Atributo: observacao
Domínio: string de caracteres ASCII

Visibilidade: privada
Tamanho: 255 caracteres
Descrição: Campo destinado a observações que o usuário achar necessária.

Classe: Despesa

Descrição: esta classe será responsável por gerenciar as contas de despesas do usuário. Cada usuário pode ter inúmeras contas de despesas desde que cada uma tenha um identificador de título único.

Atributo: tipoDespesa
Domínio: enumeração - TipoDespesa
Visibilidade: privada
Tamanho: 2 dígito
Descrição: Listagem dos possíveis títulos que uma conta despesa pode ter, por exemplo, luz, gás, telefone, etc. Este atributo será selecionado dentre um conjunto de títulos pré-disponibilizado ao usuário com a opção de inserção de um novo título.

Classe: Receita

Descrição: esta classe será responsável por gerenciar as contas de receita do usuário. Cada usuário pode ter inúmeras contas de receita desde que cada uma tenha um identificador de título único.

Atributo: tipoReceita
Domínio: enumeração - TipoReceita
Visibilidade: privada
Tamanho: 2 dígito
Descrição: Listagem dos possíveis títulos que uma conta receita pode ter, por exemplo, salário, pró-labore, aluguel, etc. Este atributo será selecionado dentre um conjunto de títulos pré-disponibilizado ao usuário com a opção de inserção de um novo título.

Classe: Investimento

Descrição: esta classe será responsável por gerenciar as contas de investimento do usuário. Cada usuário pode ter inúmeras contas de investimento desde que cada uma tenha um identificador de nome único.

Atributo: taxaIOF
Domínio: inteiro
Visibilidade: privada
Tamanho: 4 dígitos
Descrição: Valor da taxa de IOF (Imposto sobre Operações Financeiras) que será atributo a uma conta.

Atributo: taxaCPMF
Domínio: inteiro
Visibilidade: privada
Tamanho: 4 dígitos

Descrição: Valor da taxa de CMPF (Contribuição Provisória sobre Movimentação Financeira) que será atributo a uma conta.

Atributo: tipoIndiceReferencia
Domínio: enumeração - TipoIndiceReferencia
Visibilidade: privada
Tamanho: 2 dígito
Descrição: Listagem dos índices de referência que serão aplicados a uma conta.

Atributo: incideImpostoRenda
Domínio: verdadeiro/falso
Visibilidade: privada
Tamanho: booleano
Descrição: Indicador se nesta conta investimento incidirá imposto de renda. Caso verdadeiro (1, true, sim) indica que a conta terá incidência de IR. Caso falso (0, false, não) indica que não haverá incidência de IR.

Classe: Fundo

Descrição: classe abstrata que especializa os diversos tipos de fundos de investimentos possíveis que o sistema irá controlar.

Atributo: longoPrazo
Domínio: verdadeiro/falso
Visibilidade: privada
Tamanho: booleano
Descrição: Indicar se o fundo é de longo prazo ou não. Caso verdadeiro (1, true, sim) indica que o fundo é de longo prazo e assim a tabela completa de IR de ser usada. Caso falso (0, false, não) indica que o fundo é de curto prazo e devemos usar a tabela reduzida de IR.

Atributo: taxaAdministracao
Domínio: inteiro
Visibilidade: privada
Tamanho: 3 dígitos
Descrição: Percentual de administração cobrado anualmente em um fundo.

Atributo: taxaPerformance
Domínio: inteiro
Visibilidade: privada
Tamanho: 3 dígitos
Descrição: Percentual de desempenho de um fundo. A taxa de desempenho está relacionada com a possibilidade de alavancagem ou não do fundo, quanto esta alavancagem ocorre existe uma taxa de desempenho irá incidir sobre o valor de ganho acima do índice contratado pelo fundo.

Atributo: tipoFundo
Domínio: enumeração - TipoFundo
Visibilidade: privada
Tamanho: 2 dígito

Descrição: Listagem dos tipos de fundos de investimentos que serão aplicados a uma conta fundos.

Classe: CDB

Descrição: classe que conterá os CDB do usuário.

Atributo: taxaContratacao
Domínio: inteiro
Visibilidade: privada
Tamanho: 3 dígitos
Descrição: Percentual do índice de contratação de um CDB.

Atributo: dataVencimento
Domínio: data
Visibilidade: privada
Tamanho: 6 dígitos
Descrição: Data do vencimento do CDB (formato: dd/mm/aaaa).

Classe: Poupanca

Descrição: classe que conterá as cadernetas de poupança do usuário.

Classe: Acoes

Descrição: classe que conterá as ações adquiridas pelo usuário diretamente na bolsa.

Atributo: taxaCorretagem
Domínio: inteiro
Visibilidade: privada
Tamanho: 5 dígitos
Descrição: É a taxa, em percentual, de corretagem cobrada pela corretora de valores mobiliários.

Atributo: taxaEmolumentos
Domínio: inteiro
Visibilidade: privada
Tamanho: 5 dígitos
Descrição: É a taxa, em percentual, de emolumentos cobrada pela BOVESPA.

Atributo: taxaCustodiaCBLC
Domínio: inteiro
Visibilidade: privada
Tamanho: 5 dígitos
Descrição: É a taxa, em percentual, de custódia cobrada pela CBLC.

Atributo: taxaCustodiaBanco
Domínio: inteiro
Visibilidade: privada
Tamanho: 5 dígitos

Descrição: É a taxa, em percentual, de custódia cobrada pelo banco comercial.

Classe: TitulosPublicos

Descrição: classe que conterá os títulos públicos adquiridos pelo usuário.

Atributo: taxaCustodiaCBLC
Domínio: inteiro
Visibilidade: privada
Tamanho: 5 dígitos
Descrição: É a taxa, em percentual, de custódia cobrada pela CBLC.

Atributo: taxaCustodiaBanco
Domínio: inteiro
Visibilidade: privada
Tamanho: 5 dígitos
Descrição: É a taxa, em percentual, de custódia cobrada pelo banco comercial.

Atributo: tipoTituloPublico
Domínio: enumeração - TipoTituloPublico
Visibilidade: privada
Tamanho: 2 dígitos
Descrição: Listagem dos tipos de títulos públicos que serão aplicados a uma conta de título público.

Classe: Lancamento

Descrição: classe responsável por gerenciar os lançamentos feitos em uma conta. Cada lançamento está obrigatoriamente associado a um meio de pagamento e eventualmente a uma quitação.

Atributo: valor
Domínio: inteiro
Visibilidade: privada
Tamanho: 9 dígitos
Descrição: Valor de um lançamento feito em uma conta (formato: 0.000.000,00)

Atributo: inCredito
Domínio: verdadeiro/falso
Visibilidade: privada
Tamanho: booleano
Descrição: Indicador se o lançamento é de crédito ou de débito. Se o lançamento é de crédito o valor será true = 0 = sim, se o lançamento for de débito o valor será false = 1 = não.

Atributo: dataVencimento
Domínio: data
Visibilidade: privada
Tamanho: 6 dígitos
Descrição: Data de vencimento do lançamento (formato: dd/mm/aaaa).

Atributo: dataQuitacao
Domínio: data
Visibilidade: privada
Tamanho: 6 dígitos
Descrição: Data da quitação do lançamento (formato: dd/mm/aaaa).

Atributo: tipoMeioPagamento
Domínio: enumeração - TipoMeioPagamento
Visibilidade: privada
Tamanho: 2 dígitos
Descrição: Listagem dos tipos de meio de pagamento utilizados no lançamento.

Abaixo temos o a listagem das enumerações (enumerations) utilizadas no sistema com as suas descrições.

Enumeração: Idioma

Descrição: esta enumeração terá a função de armazenar os possíveis idiomas que um usuário pode utilizar no sistema. Cada usuário terá um idioma associado a ele.

Atributo: nome
Domínio: string de caracteres ASCII
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Nome do idioma. O sistema será desenvolvido para suportar diversos idiomas, podendo o usuário utilizar o que melhor lhe convir.

Atributo: numero
Domínio: inteiro
Visibilidade: privada
Tamanho: 3 dígitos
Descrição: Número do idioma. O número do idioma é um código internacional do idioma, por exemplo, 151 é Português Brasil, 56 é Inglês EU, etc.

Enumeração: TipoDespesa

Descrição: esta enumeração conterá os diversos tipos de despesas que uma conta despesa pode referenciar.

Atributo: nome
Domínio: string de caracteres ASCII
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Nome do tipo da despesa.

Enumeração: TipoReceita

Descrição: esta enumeração conterá os diversos tipos de receitas que uma conta receita pode referenciar.

Atributo: nome
Domínio: string de caracteres ASCII
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Nome do tipo da receita.

Enumeração: TipoIndiceReferencia

Descrição: esta enumeração conterá os diversos tipos de índice de referência que uma conta investimento pode ter.

Atributo: nome
Domínio: string de caracteres ASCII
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Nome do tipo do índice de referência.

Enumeração: TipoTituloPublico

Descrição: esta enumeração conterá os diversos tipos de títulos públicos que uma conta título público pode ter.

Atributo: nome
Domínio: string de caracteres ASCII
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Nome do tipo de título público.

Enumeração: TipoFundo

Descrição: esta enumeração conterá os diversos tipos fundos que uma conta fundo pode ter.

Atributo: nome
Domínio: string de caracteres ASCII
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Nome do tipo do fundo.

Enumeração: TipoMeioPagamento

Descrição: esta enumeração conterá os diversos tipos de meios de pagamento que um lançamento pode ter.

Atributo: nome
Domínio: string de caracteres ASCII
Visibilidade: privada
Tamanho: 20 caracteres
Descrição: Nome do tipo do meio de pagamento.

3. Projeto

3.1. Componentização utilizando UML

3.1.1. Identificação de Componentes

O objetivo desta fase é identificar um conjunto inicial de interfaces de negócio para os componentes de negócio, interface de sistemas para os componentes de sistema e montar uma arquitetura inicial de componentes. Como pode ser visto na figura abaixo processo de identificação de componentes tem como insumo para o seu desenvolvimento dois artefatos, a saber, o modelo de análise (diagrama de classes) e o modelo de casos de uso.

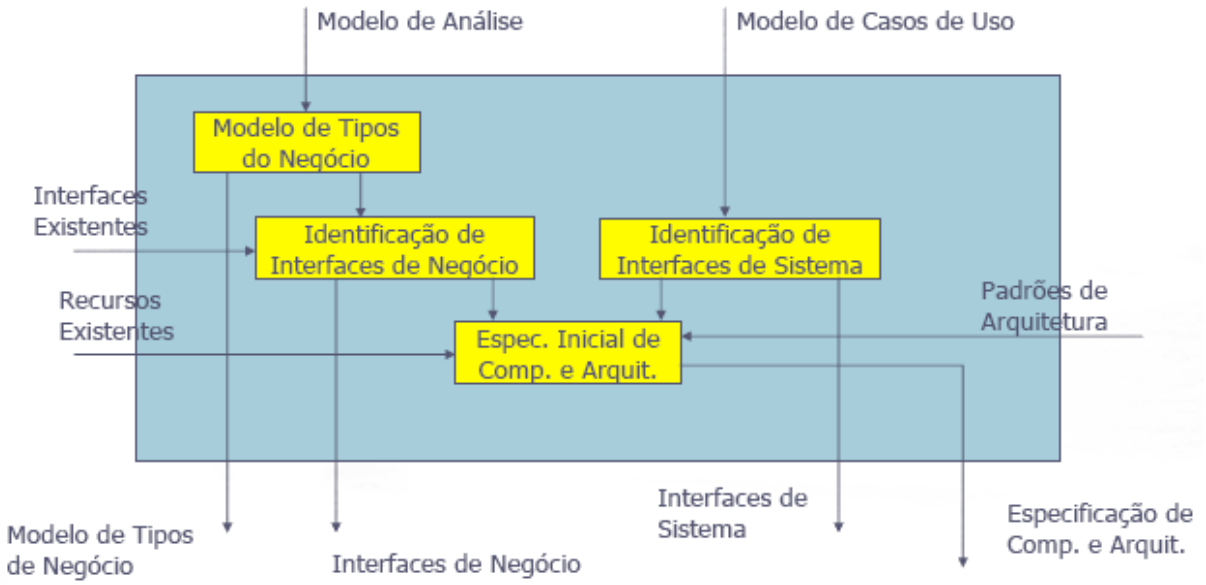

Figura 8 - Identificação de componentes

Do modelo de casos de uso iremos determinar as interfaces de sistema utilizando o processo de identificação de interfaces de sistema. Estas interfaces irão implementar a lógica do caso de uso, como um caso de uso é na verdade uma seqüência de passos, a fim de realizar uma tarefa específica, cada passo deverá ser atendido por uma ou mais operações. Sendo assim as interfaces do sistema são derivadas do modelo de casos de uso e as correspondentes operações, que irão implementar os passos do caso de uso, são derivadas das descrições dos casos de uso.

O gerenciamento da lógica do caso de uso, ou seja, a ordem com que os passos são executados, os fluxo alternativos, etc. serão contemplados no controle de diálogo com o usuário. A seguir temos um exemplo deste processo. O item 1 é um pacote que contem os casos de uso relacionados a um mesmo assunto; o item 2 é o controle do diálogo com o usuário; o item 3 é a interface propriamente dita, é nesta interface que são criados os serviços necessários a execução do caso de uso.

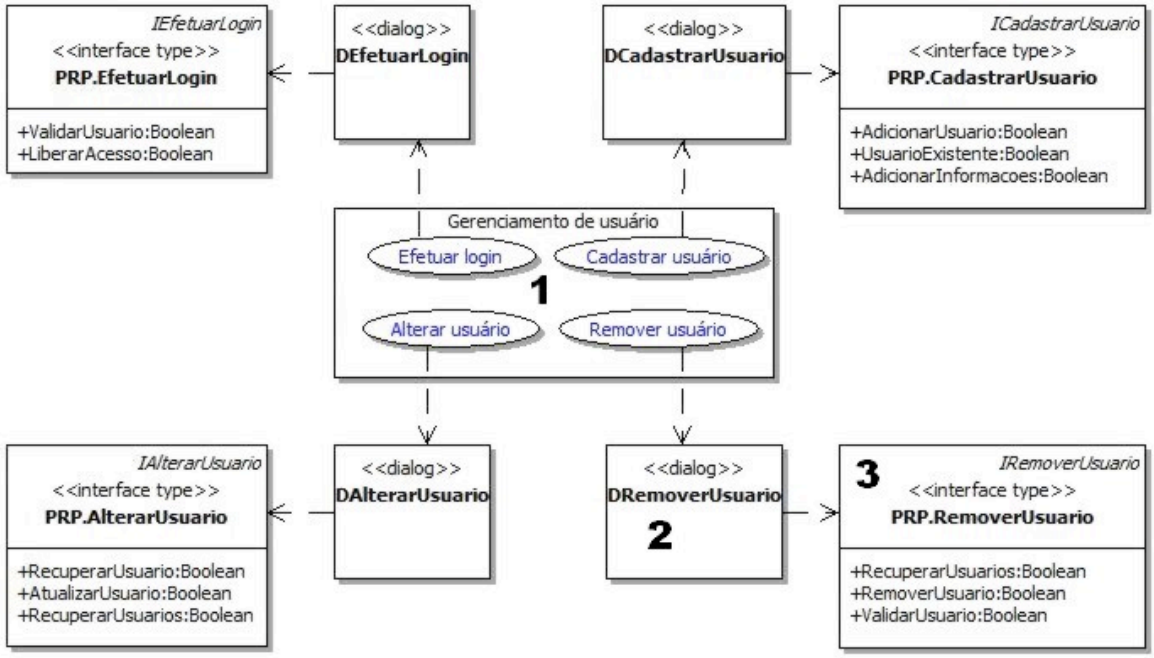

Figura 9 - Diagrama de Interfaces e Diálogo - Gerenciamento de usuário

A seguir temos os diagramas de interfaces e diálogo de cada pacote do sistema.

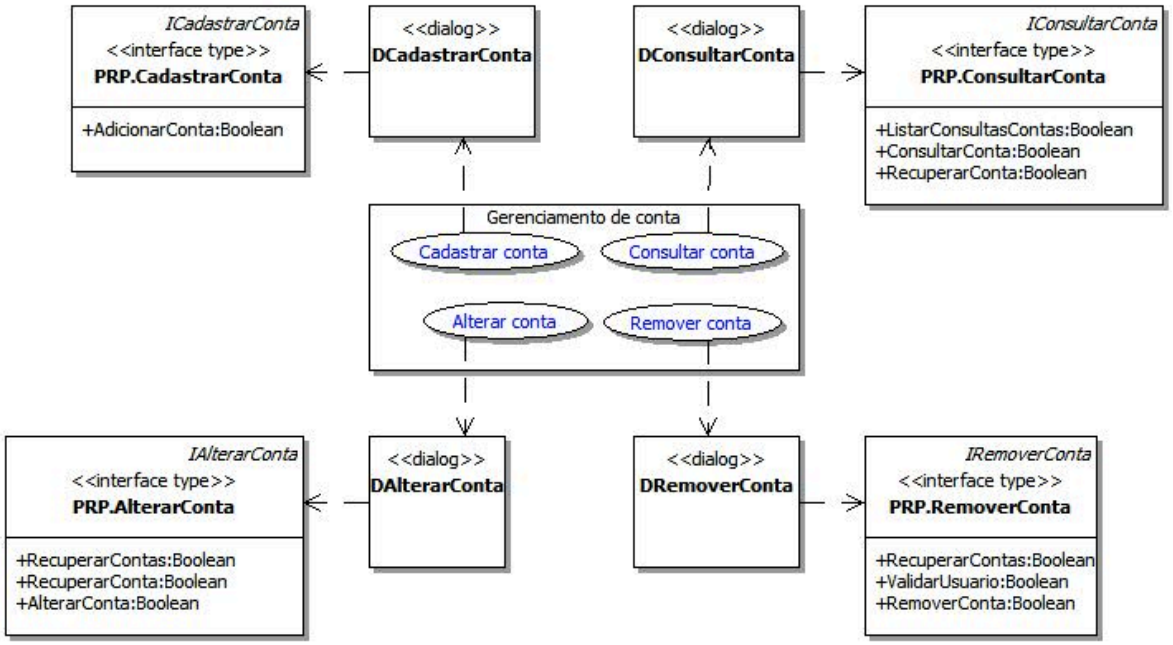

Figura 10 - Diagrama de Interfaces e Diálogo - Gerenciamento de conta

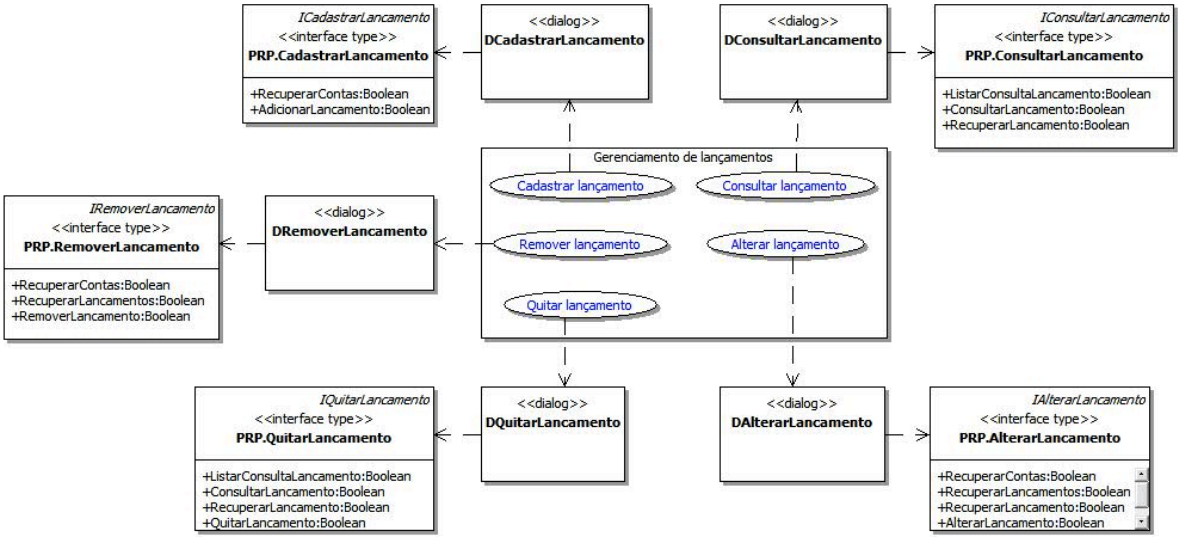

Figura 11 - Diagrama de Interfaces e Diálogo - Gerenciamento de lançamentos

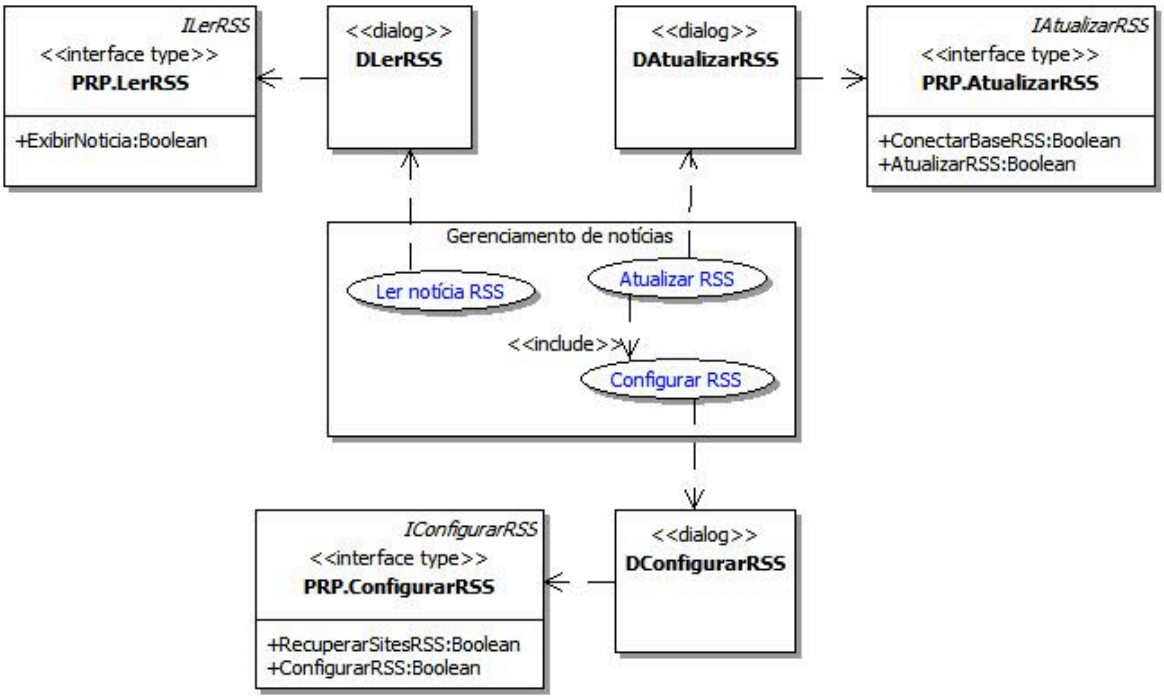

Figura 12 - Diagrama de Interfaces e Diálogo - Gerenciamento de notícias

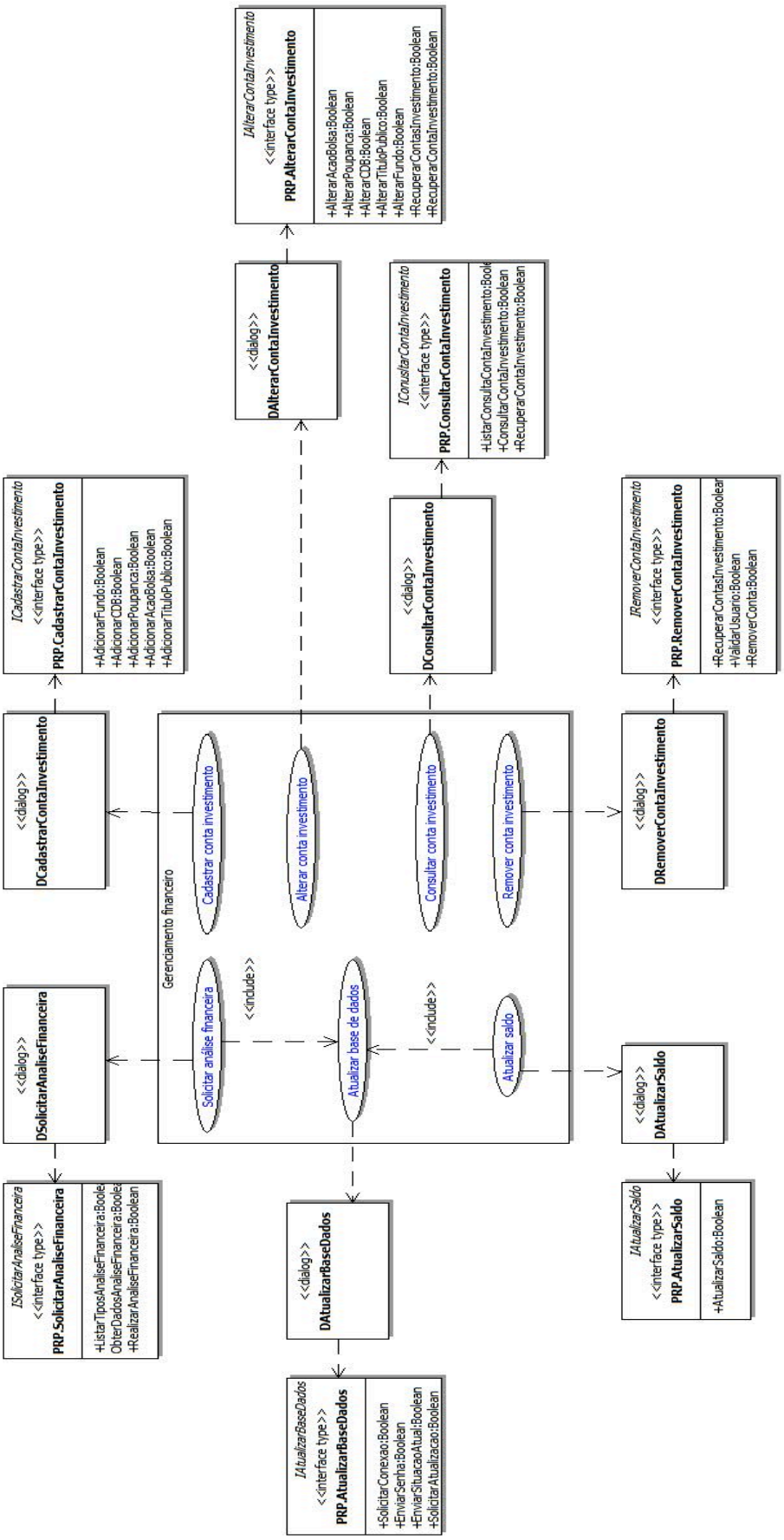

Figura 13 - Diagrama de Interfaces e Diálogo - Gerenciamento financeiro

Para a especificação das interfaces de negócio e modelo de tipos de negócio utilizamos o modelo de análise, que nada mais é do que o modelo de classes da orientação a objetos. O modelo de tipos de negócio vem a ser a visão de negócio sob a perspectiva do sistema, representando informações do negócio que devem ser mantidas. Já as interfaces de negócio são estruturas que irão gerenciar um grupo de tipos centrais do negócio. O processo consiste em, do modelo de análise identificar os tipos centrais do negócio e nomear-los com o estereótipo <<core>> e todos os outros tipos como o estereótipo <<type>>. O próximo passo é criar uma interface de negócio associada a cada tipo central identificado, ou seja, se temos 3 tipos centrais de negócio teremos 3 interfaces de negócio, uma para cada tipo.

Por exemplo, fazendo uma análise minuciosa no sistema detectamos que este possui 3 tipos centrais de negócio, que estão intimamente ligados aos requisitos do sistema. São eles; cliente, conta e analise financeira. Esta divisão além de estar em sintonia com o negócio a qual o sistema deve atender, simplifica futuras expansões e manutenções do mesmo. Neste cão teremos 3 interfaces de negócio a serem criadas; IGerenciaCliente, IGerenciaConta, IGerenciaAnaliseFinanceira. Abaixo temos o diagrama dos tipos centrais de negócios com suas respectivas classes associadas.

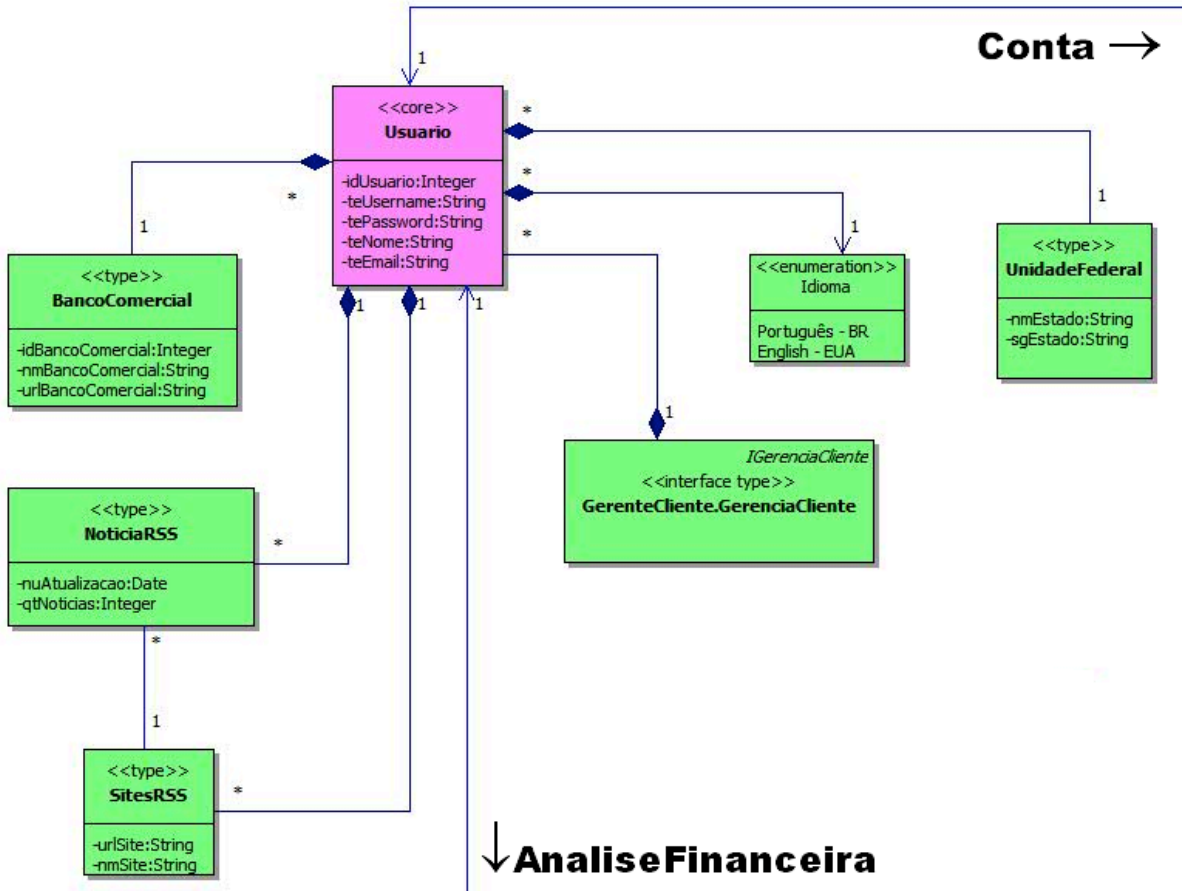

Figura 14 - Interface de negócio de Cliente

No diagrama acima devemos notar duas particularidades, uma diz respeito às associações por composição feitas entre cliente e todas as classes pertencente à interface de cliente. Esta situação é para demonstrar uma relação parte-todo com dependência, ou seja, se um cliente é excluído todos os seus dados de configuração, notícias, etc. são excluídos conjuntamente. Outro ponto a ser notado é quando a associação entre cliente e conta e cliente e analise financeira. Estas associações são navegáveis devido ao fato de termos um "ponteiro" entre as duas estruturas que vem a ser um identificador único de uma estrutura, que será armazenado na outra estrutura. Desta forma somente criamos dependência em um sentido, ou seja, cliente não tem "chave" para conta ou análise financeira e sim estas duas é que tem para cliente.

Sendo assim identificado um tipo central de negócio, o cliente, foi criado uma interface, IGerenciaCliente, que será responsável por orquestrar o negócio em conjunto com as outras interfaces. Abaixo temos o diagrama das outras interfaces de negócio.

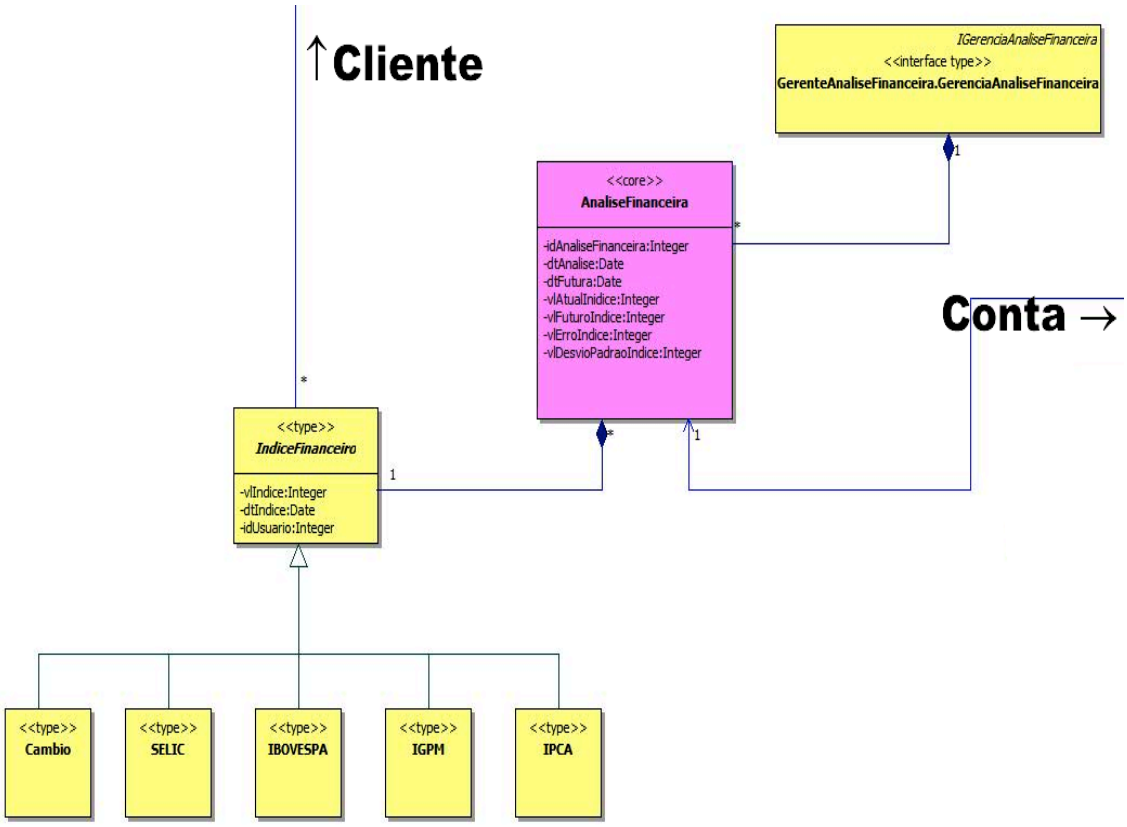

Figura 15 - Interface de negócio de Analise Financeira

Para o tipo de negócio identificado, a Análise Financeira, procedeu-se da mesma forma que para o Cliente, criando uma interface de negócio e associando as classes que possuem relação com a análise financeira por composição. Por fim abaixo temos o último tipo de negócio identificado, a Conta, com suas associações e sua interface de gerencia, IGerenciaConta.

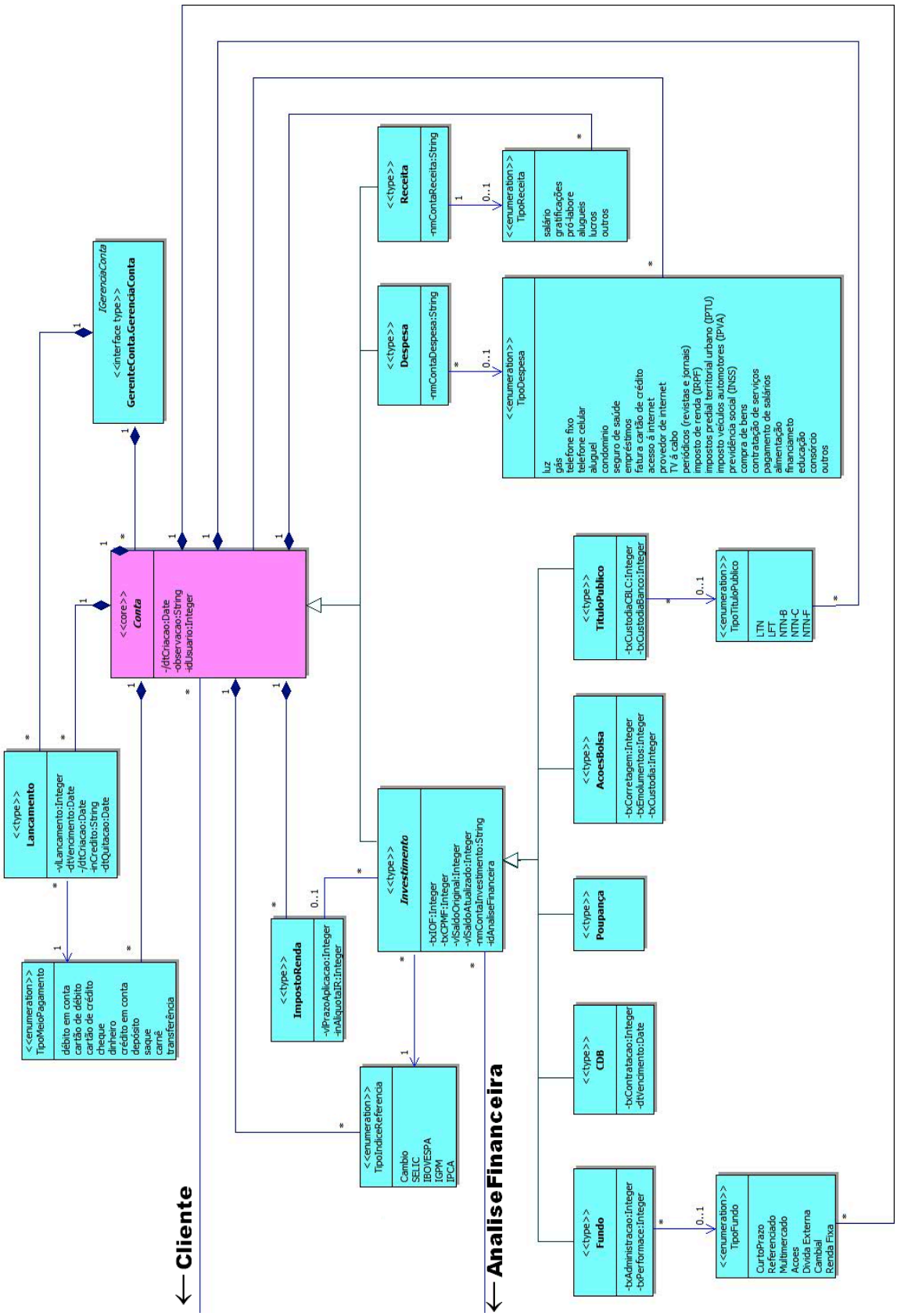

Figura 16 - Interface de negócio de Conta

Feito isso temos agora um diagrama de responsabilidade de interfaces do sistema, na próxima página.

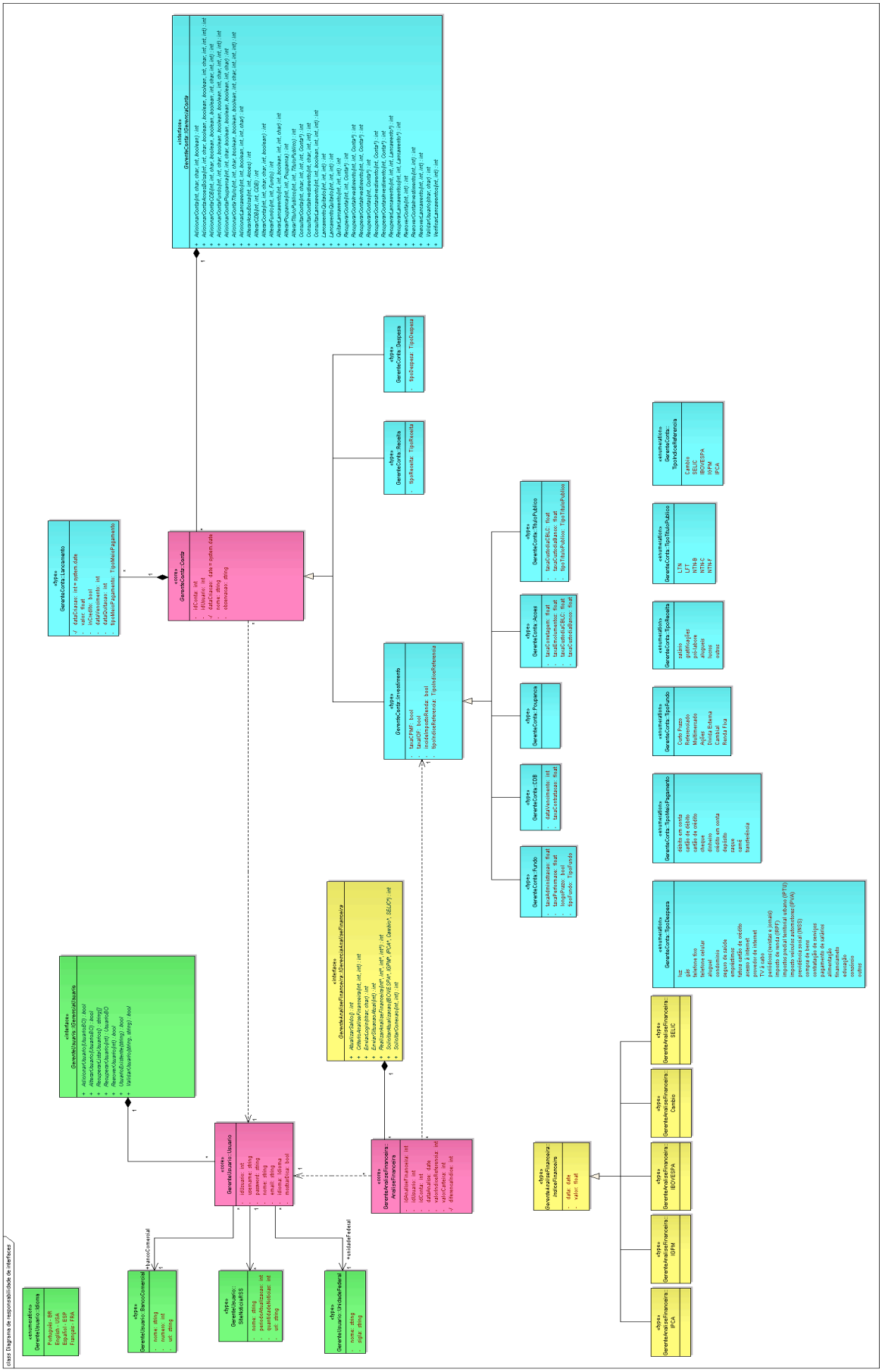

Figura 17 - Diagrama de Responsabilidade de Interfaces

Com isso terminamos a fase de identificação de componentes, tendo como produto desta fase o primeiro diagrama de componentes do sistema. Cada interface de negócio identificada irá tornar-se um componente, ou seja, teremos 3 componentes de negócio, e ainda teremos um componente de sistema que será responsável pelas interfaces do PRP com o usuário, ou seja, os casos de uso.

Os componentes de negócio serão responsáveis, cada um deles, por uma parte do sistema, a saber; clientes, contas, e análise financeira. Cada componente de negócio irá prover serviços que deverão ser orquestrados pelo componente de sistema a fim de prover as funcionalidades que os usuários irão desempenhar perante o sistema. Sendo assim temos na próxima página o diagrama de componentes do PRP.

O próximo passo será fazer a interação entre os componentes de negócio a fim de prover as funcionalidades levantadas nos casos de uso.

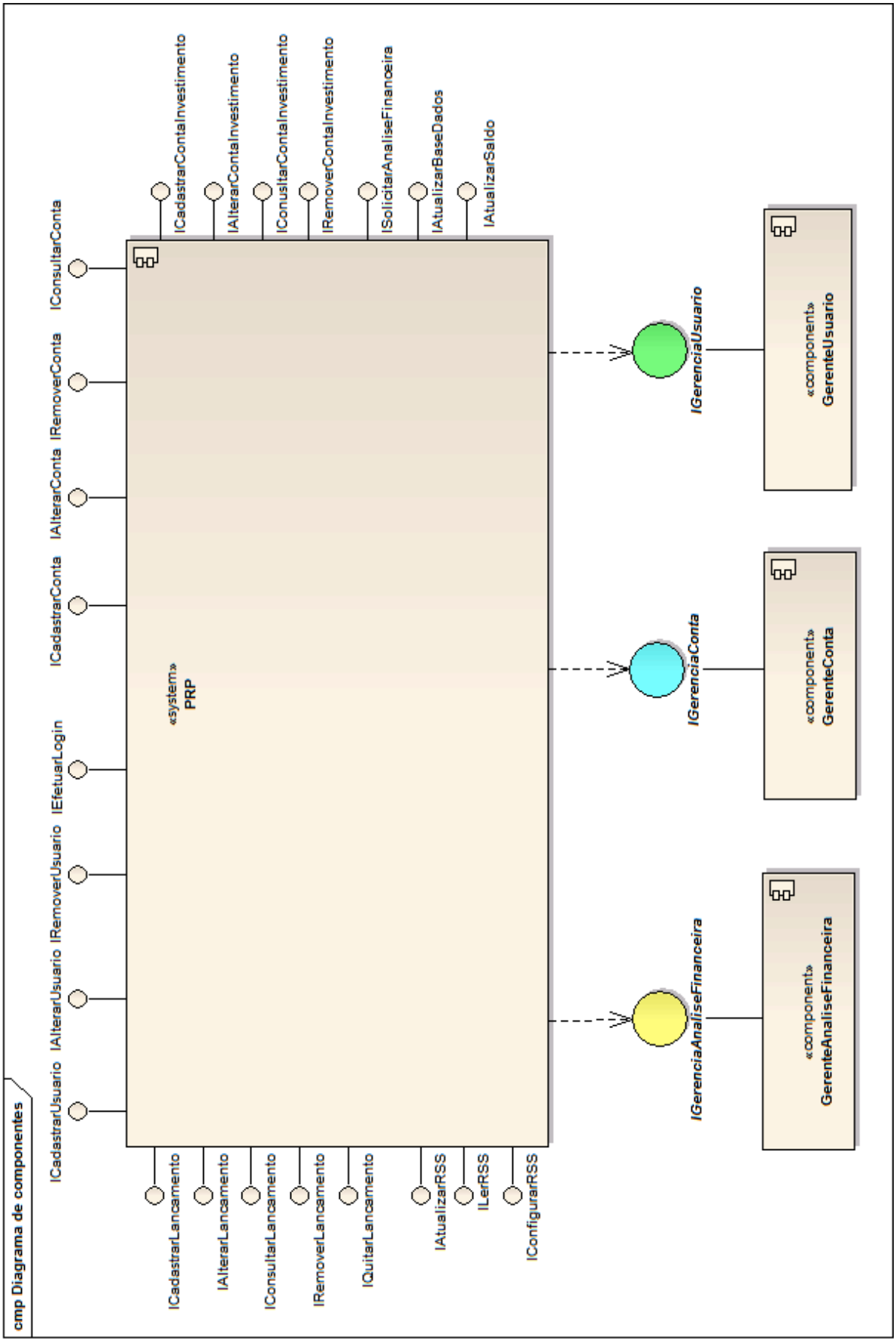

Figura 18 - Diagrama de componentes

3.1.2. Interação de Componentes

Esta fase de interação de componentes tende a ser uma fase de menor complexidade, contudo de enorme esforço, devido ao fato de ser nesta fase que iremos descobrir as operações de negócio para satisfazer os casos de uso. Estas operações serão descobertas através de diagramas de interação de componentes, ou seja, um diagrama irá nos mostrar como deve ser a interação entre o componente de sistema e entre os componentes de negócio a fim de produzir serviços para atender ao funcionamento do sistema.

Durante o processo de definição dos serviços iremos também determinar os parâmetros e estruturas de dados necessários as operações. Na interação de componentes podemos chegar a fazer um refinamento das interfaces, unido algumas, separando outras, enfim podemos reorganizar a divisão das interfaces, ou seja, fazer um refinamento. E caso este refinamento ocorra podemos ter que fazer também um refinamento na arquitetura inicial dos componentes.

Agora vamos iniciar a interação dos componentes para descobrir os serviços que iremos ter que implementar. Iremos fazer um diagrama de interação de componentes para cada necessidade levantada nas interfaces do sistema, ou seja, os passos do caso de uso.

Devido ao fato de serem diversos os diagramas de interação, inicialmente um para cada caso de uso desenvolvido durante a fase de levantamento, tomei a decisão de somente explicar o seu funcionamento, acima, tendo os diagramas em si representados na ferramenta de modelagem. Esta facilidade foi possível visto que a ferramenta gera um relatório em formato HTML com toda modelagem feita, neste caso tais relatórios estão no CD a ser entregue juntamente com este projeto.

3.1.3. Especificação de Componentes

Esta fase de como produto base a especificação completa das operações do sistema, a saber, entradas, saídas, restrições e mudanças de estados, ou seja, é o contrato das interfaces. Este contrato irá determinar como utilizar a interface, nele irão constar os resultados esperados dado condições específicas de entradas. Na indústria de software orientado a

componentes usa-se o termo, *Design by Contract*. Como este termo não tem uma tradução que ilustre seu real significado iremos dar aqui uma descrição da sua finalidade.

Desgin by Contract vem a ser uma especificação das pré e pós-condições de cada operação do sistema.

Nesta fase também iremos introduzir as regras do negócio, ou seja, os requisitos levantados devem ser introduzidos nesta parte da modelagem na forma de regras de negócio. Por exemplo, se um sistema deve realizar uma inclusão de um item dado certas condições (definidas pelo negócio ao qual o sistema irá representar), estas regras irão ser traduzidas em pseudocódigos, ou até mesmo códigos e algoritmos, neste momento.

Para termos um alinhamento das definições utilizadas nesta fase temos que; pré-condição – descreve tudo o que deve ser verdadeiro antes da operação ser iniciada, para que a operação garanta a pós-condição. Já pós-condição é o resultado da operação, caso tenha sido verdadeira a pré-condição. Vemos que pré e pós-condição estão intimamente ligadas, visto que uma somente pode existir de fato caso a outra seja atendida.

Abaixo temos como ficaram modelados os três componentes apresentados no diagrama de componentes na página 62.

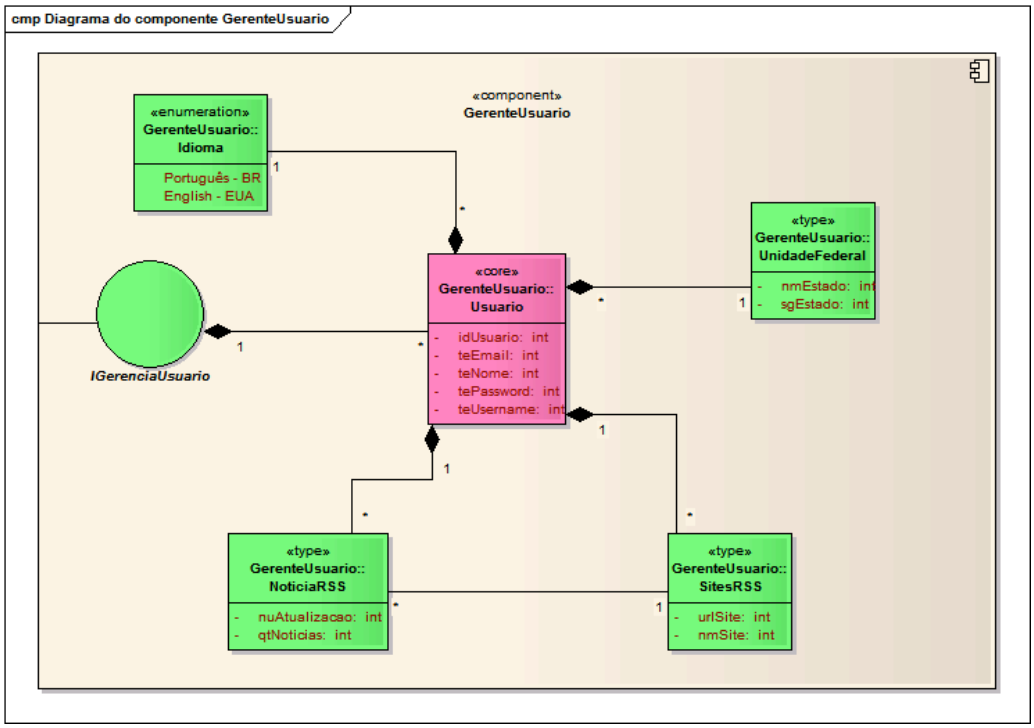

Figura 19 - Diagrama do componente GerenteUsuario

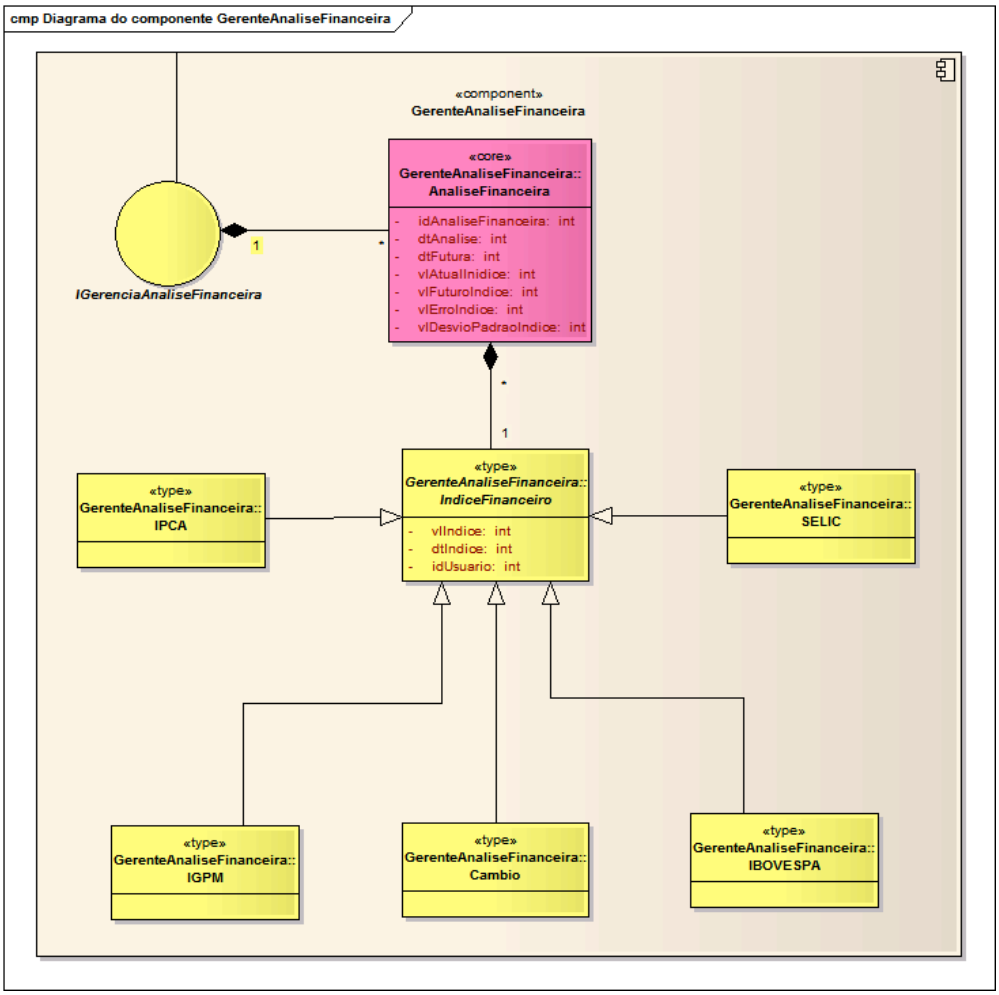

Figura 20 - Diagrama do componente GerenteAnaliseFinanceira

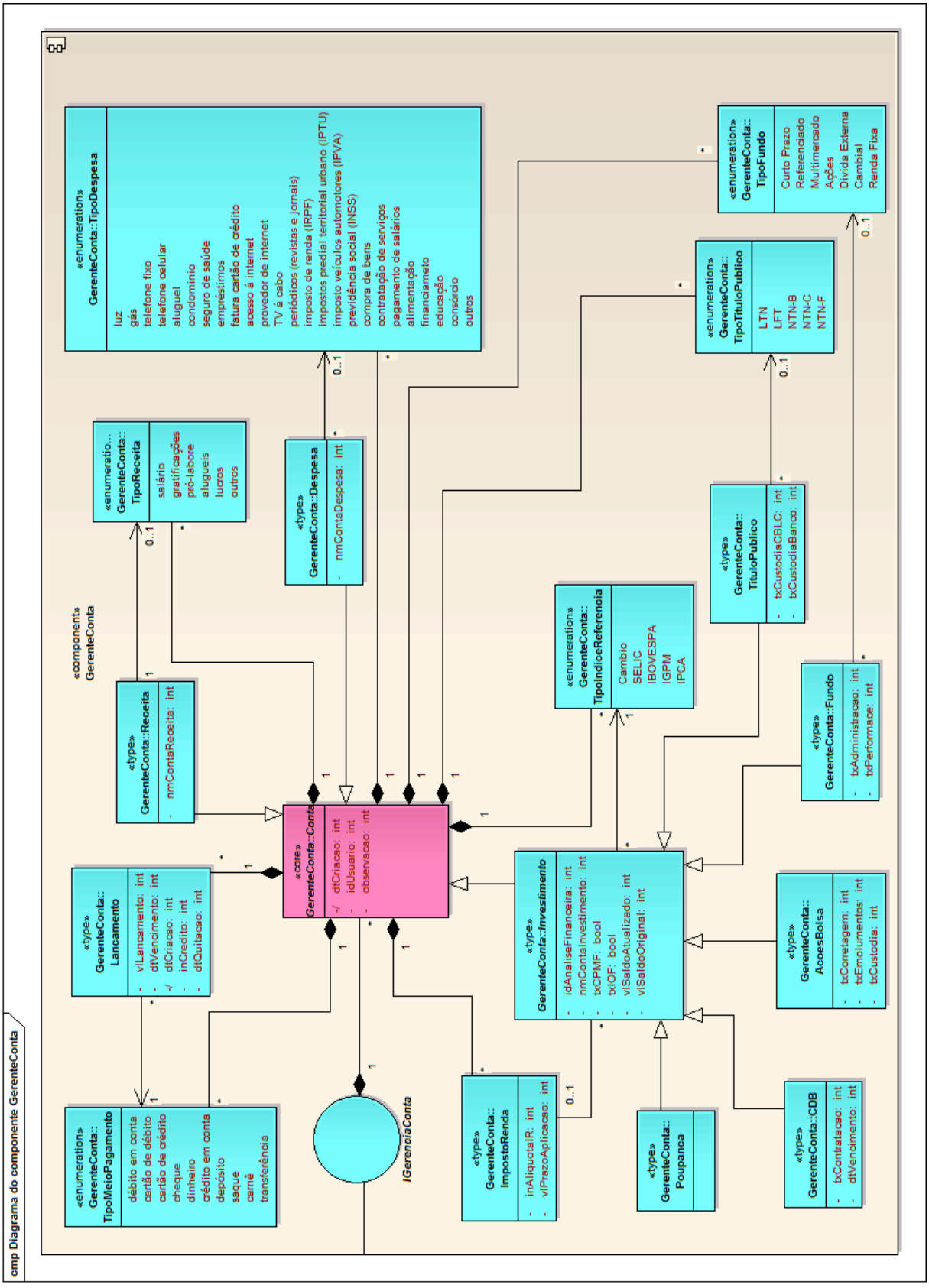

Figura 21 - Diagrama do componente GerenteConta

3.2. Modelagem objeto relacional

Neste momento já temos condições de fazer o modelo objeto relacional que será responsável pela persistência dos dados do sistema. Como visto anteriormente o PRP terá 3 componentes, o de cliente, o de conta e o de análise financeira. A componentização com UML prega que os dados devem fazer parte do componente, ou seja, as estruturas de dados que irão subsidiar o armazenamento dos dados estão "dentro" dos componentes, sendo assim cada componente tem a sua estrutura de dados.

Com visto no diagrama de responsabilidade de interface a forma de "unir" os componentes do sistema será utilizando um identificador único entre eles. Este identificador não precisa ser obrigatoriamente a chave primária em uma estrutura de dados, contudo se a utilizarmos desta forma aumentamos a integridade dos dados. Neste caso o componente cliente terá um identificador na classe cliente (idUsuario) que será "exportado" para as classes, conta e analise financeira, que pertence, respectivamente, ao componente conta e componente análise financeira.

3.2.1. Componente Usuário

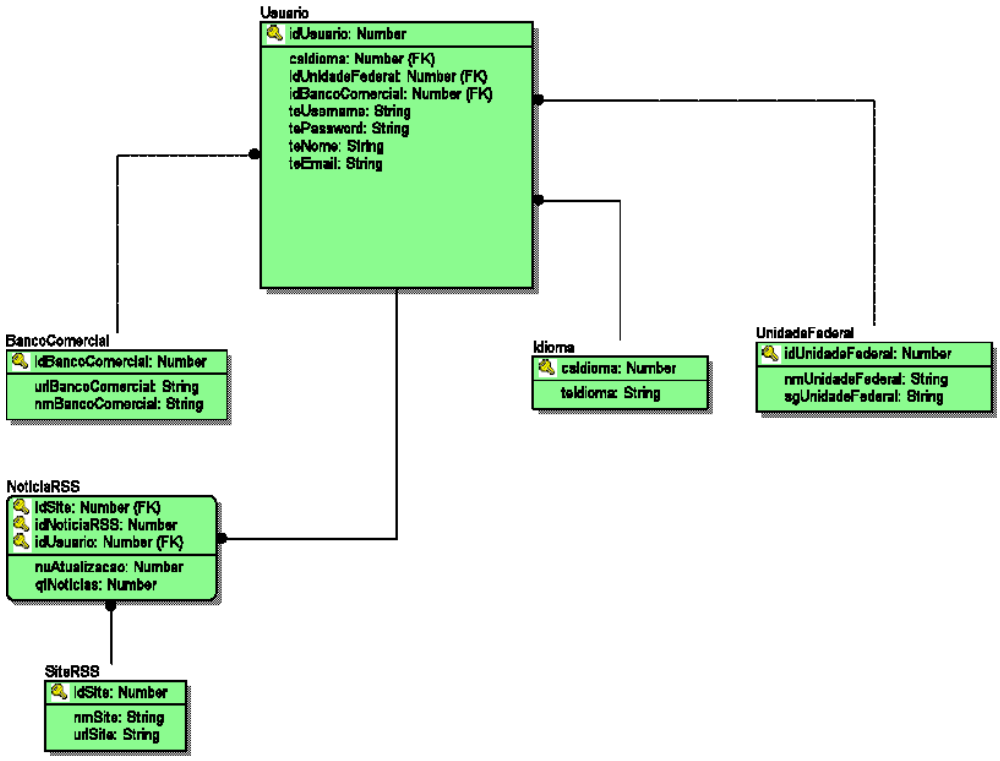

Figura 22 - Modelagem Objeto Relacional do Componente Usuário

3.2.2. Componente Conta

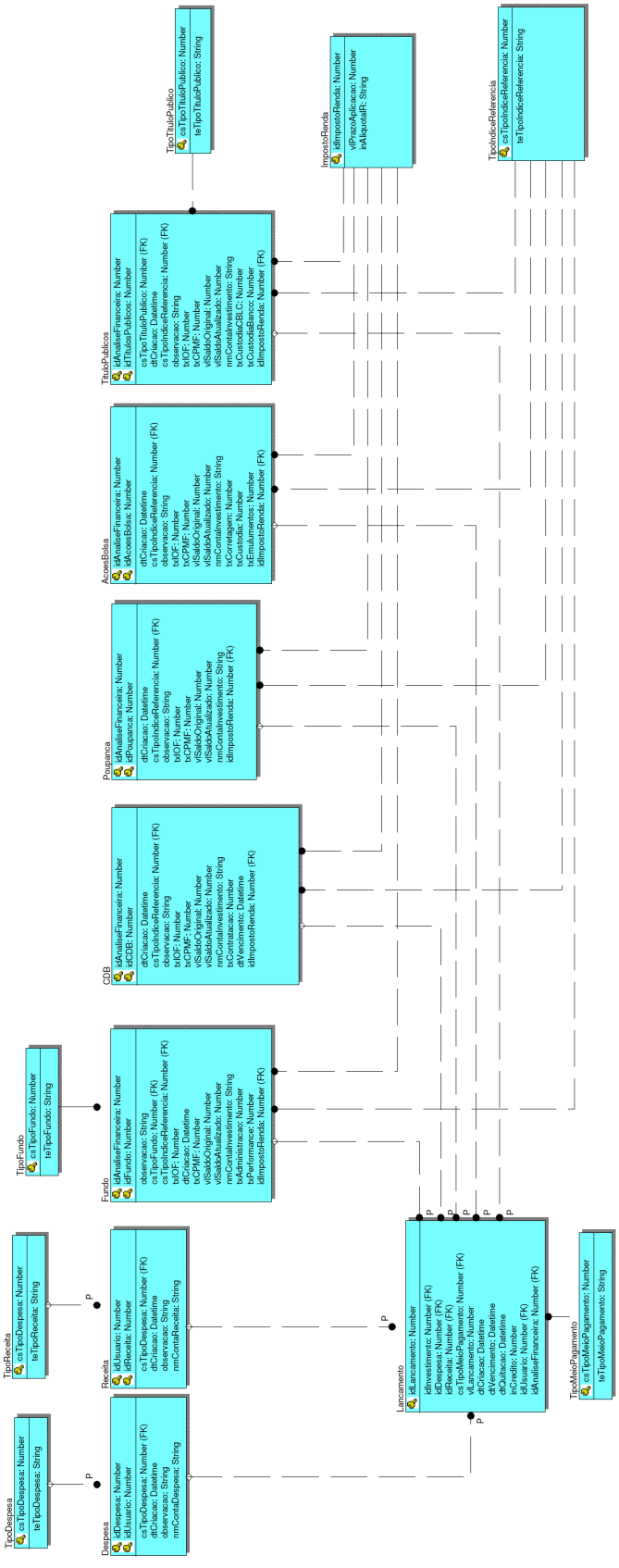

Figura 23 - Modelagem Objeto Relacional do Componente Conta

3.2.3. Componente Análise Financeira

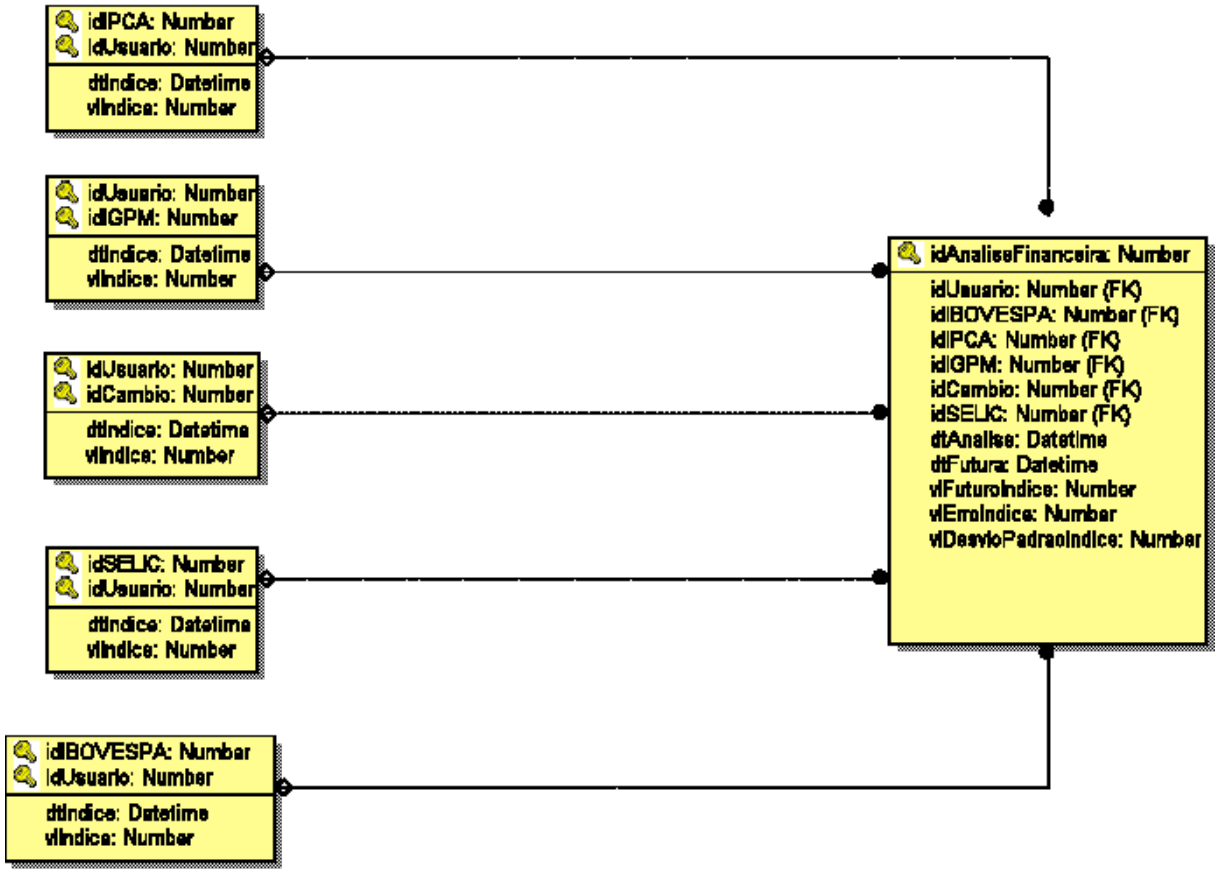

Figura 24 - Modelagem Objeto Relacional do Componente Analise Financeira

4. Desenvolvimento

4.1. Desenvolvimento

O desenvolvimento do projeto foi dividido em fases que seriam implementadas ao longo do tempo. A fase que foi escolhida para ser implementada neste projeto final de curso foi a Fase 1 que consistiu em criar uma estrutura geral do sistema que futuramente receberia as outras partes para assim ir aumentando seu número de funcionalidades e qualidade na resposta a que o sistema se propõem. Digo isso devido ao fato de elaborar um sistema de apoio a decisões demonstrar um grau de complexidade altíssimo, que certamente fugiria ao escopo do projeto, além da necessidade de estudos mais aprofundados e troca de experiências com outros profissionais.

A IDE utilizada foi o Eclipse, visto que a mesma possui diversos módulos/plug-in que fazem do seu uso obter significativos ganhos de produtividade, alem é claro, de ser a IDE mais utilizada no mercado de desenvolvimento de software. Em específico a biblioteca utilizada, wxWidgets, deve que ser recopilada para "trabalhar" mais acoplada ao Eclipse.

A forma de desenvolvimento adotada foi a de ir construindo os casos de uso, utilizando inicialmente os mais simples para validar a arquitetura (framework) do projeto e depois ir construindo os casos de uso mais complexos. Esta abordagem mostrou-se favorável a possíveis mudanças, inclusive colocada-a em teste, quando mudanças consideráveis foram feitas no projeto.

A arquitetura em 3 camadas, MVC, mostrou-se a melhor escolha em vista do desacoplamento entre apresentação, modelo e dados, facilitando assim mudanças nestas camadas da aplicação com o menor impacto possível, ou até mesmo nenhum impacto, sempre respeitando os "contratos" utilizados para o acoplamento das camadas.

Nesta arquitetura em 3 camadas, a camada de apresentação foi toda desenvolvida utilizando-se a biblioteca wxWidgets com seus componentes "pré-construídos" (janelas, botões, menus, etc.) Na camada de inteligência do sistema, a camada de negócio, foi utilizado a linguagem C++ que foi a base escolhida para o projeto e por fim na camada de dados foi utilizado a biblioteca SQLite que é um SGBD baseado em arquivo.

Abaixo temos uma explicação da estrutura em camadas do PRP.

Arquitetura em 3 camadas, baseada no modelo MVC, onde temos a camada de apresentação utilizando a biblioteca wxWidgets. Na camada de negócio utilizamos a linguagem C++, e por fim, a camada de dados, onde utilizamos a biblioteca wxSQLite 3, baseada na biblioteca SQLite3, que implementa uma solução de SGBD em arquivo.

Figura 25 – Arquitetura em 3 camadas

O funcionamento das camadas é mos trado abaixo:

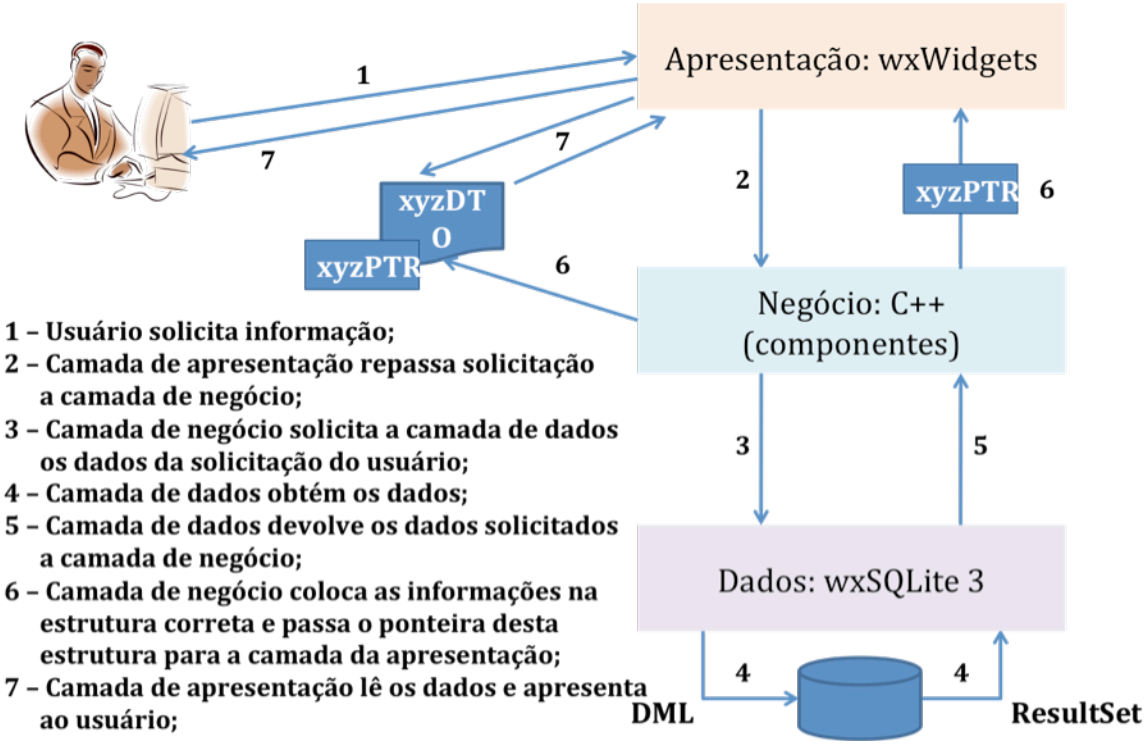

Figura 26 – Execução da arquitetura

Na figura abaixo temos a arquitetura sendo executada em um formado de diagrama de seqüência, estrutura esta que foi sempre seguida no projeto.

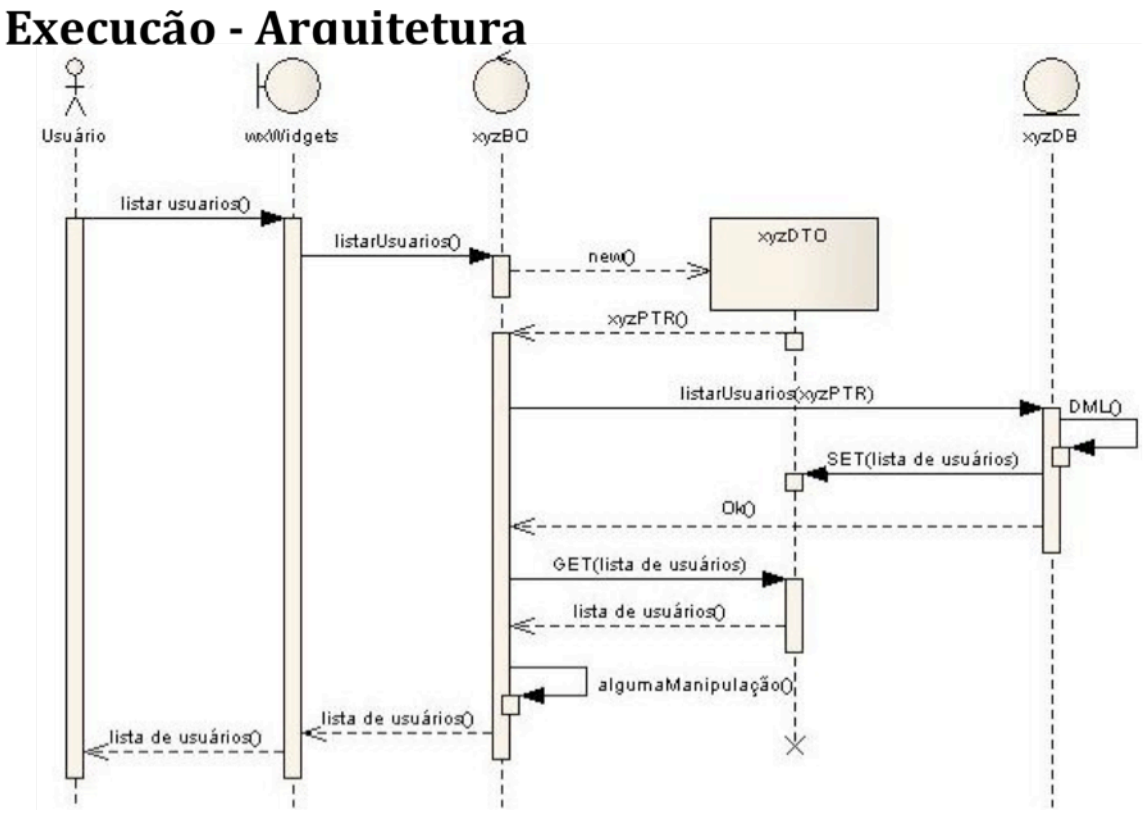

Figura 27 – Execução da arquitetura – Diagrama de seqüência

4.2. Arquitetura

Como dito no item anterior a arquitetura utilizada foi a em 3 camadas. Para esta estrutura ser de fácil entendimento, e para futuras ampliações das funcionalidades do PRP foi utilizada uma técnica de distribuição de arquivos muito utilizada em projeto Web-Based que usam J2EE, a saber, cada arquivo fonte do projeto tem somente uma classe que desempenha um específico papel dentro do sistema todo.

Esta forma de codificação mostra-se muito boa para manter o maior desacoplamento possível entre as diversas classes do projeto, por exemplo, se uma classe de negócio precisa de um resultado de dados (acesso ao SGBD) ela simplesmente instancia a classe de conexão e pede o resultado a esta instancia da classe de conexão ao SGBD. As necessidades de conexão, as estruturas de DML (select), estruturas de resultados de consultas, enfim tudo necessário ao acesso e recuperação de informações do SGBD está na classe de conexão.

Nesta estrutura as classes de apresentação somente são responsáveis por apresentar os diálogos com o usuário, obter os dados e retornar resultados ou respostas ao usuário, ou seja, as classes de apresentação não sabem nada de negócio e de acesso aos dados. Para a

interligação entre as diversas camadas foi utilizado o conceito de objeto de transferência, que no caso deste projeto é o objeto de negócio, um espelho da estrutura dos dados que está armazenada no SGBD.

Vamos a um exemplo do funcionamento desta arquitetura. Para uma ilustração completa do problema vou utilizar uma função de alteração de dados do usuário (caso de uso alterar usuário), visto que nesta função teremos tanto a recuperação dos dados do usuário como a sua alteração e posterior armazenagem dos novos dados.

Na figura 28 podemos ver a estrutura de classes de diálogo e de gerencia (controladores do negócio). Associado ao caso de uso Alterar Usuário há uma classe diálogo (DialogoUsuario) que implementa toda a interface gráfica com o usuário necessária pra a execução das tarefas referentes a manutenção do usuário (incluir, alterar, login, etc...). Esta classe, para o nosso exemplo, tem o método Update() que tem a finalidade de "montar" a interface gráfica que será necessária para executar a alteração dos dados do usuário e "chamar" os outros métodos que serão responsáveis por recuperar/gravar os dados.

Na classe de negócio, IGerenciaUsuario, existem dois métodos que serão utilizados no nosso exemplo, o RecuperarUsuario(int):UsuarioBO e o AlterarUsuario(UsuarioBO):bool. Neste ponto a classe UsuarioBO (objeto de transferência) aparece tanto como saída de um método como entrada de outro. Vemos que a classe UsuaioBO não possui nada além de atributos que irão transportar os dados do usuário de uma camada para outra, alem é claro, dos métodos get() e set() referentes a cada atributo. Vamos a um passo a passo mais claro para ilustrar o funcionamento:

1 – Usuário seleciona alterar usuário no meu de usuário;

2 – O frame principal do PRP captura a requisição e aciona o método Update() da classe DialogoUsuario;

3 – O método Update() monta o dialogo de alteração e solicita a recuperação dos dados do usuário através do acesso a classe IGerenciaUsuario usando o método RecuperarUsuario();

4 – A classe IGerenciaUsuario obtém o id do usuário conectado e acessa o banco de dados através da classe UsuarioDB usando o método getUsuario(id);

5 – O resultado da consulta (resultset) é colocado no objeto de transferência UsuarioBO e retornado para a classe de gerenciamento de usuário;

6 – O gerente de usuário repassa o BO para a classe de dialogo que por sua vez preenche os campos com os dados do usuário e apresenta o dialogo;

7 – Neste ponto o usuário altera seus dados e confirma (botão Ok);

8 – A classe de diálogo recupera os dados informados e os armazena no objeto de transferência UsuarioBO e executa o método AlterarUsuario(UsuarioBO) da classe de gerencia de usuário;

9 – A classe de gerencia solicita que os novos dados de usuário sejam gravados executando o método gravarUsuario(UsuarioBO) da classe UsuarioDB;

10 – A classe UsuarioBO abre a conexão com o banco e grava os dados respondendo Ok para a camada superior. Este Ok vai se propagando até a camada de apresentação que finalmente informa ao usuário que as alterações foram executadas com sucesso.

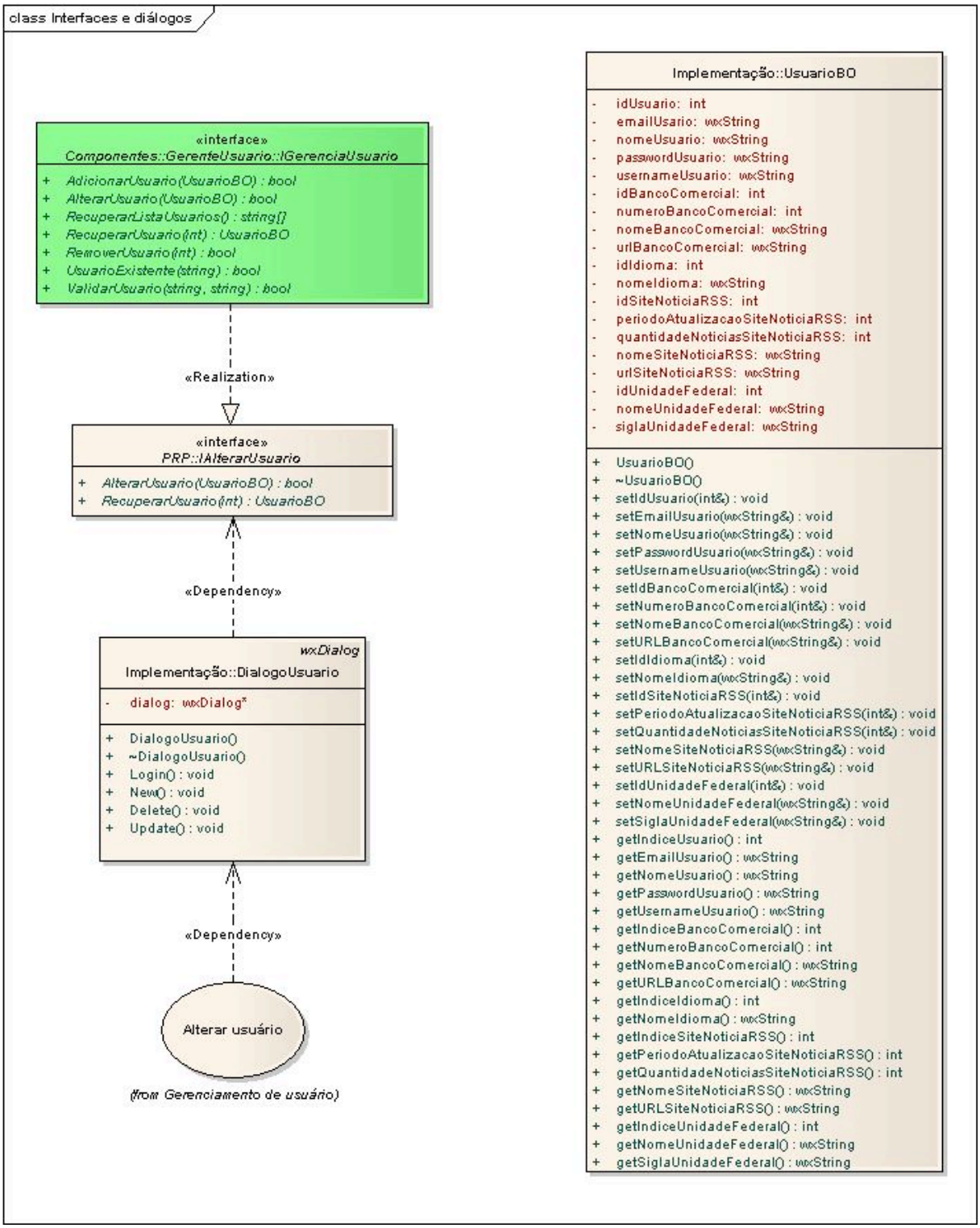

Figura 28 – Classe do Altera Usuário

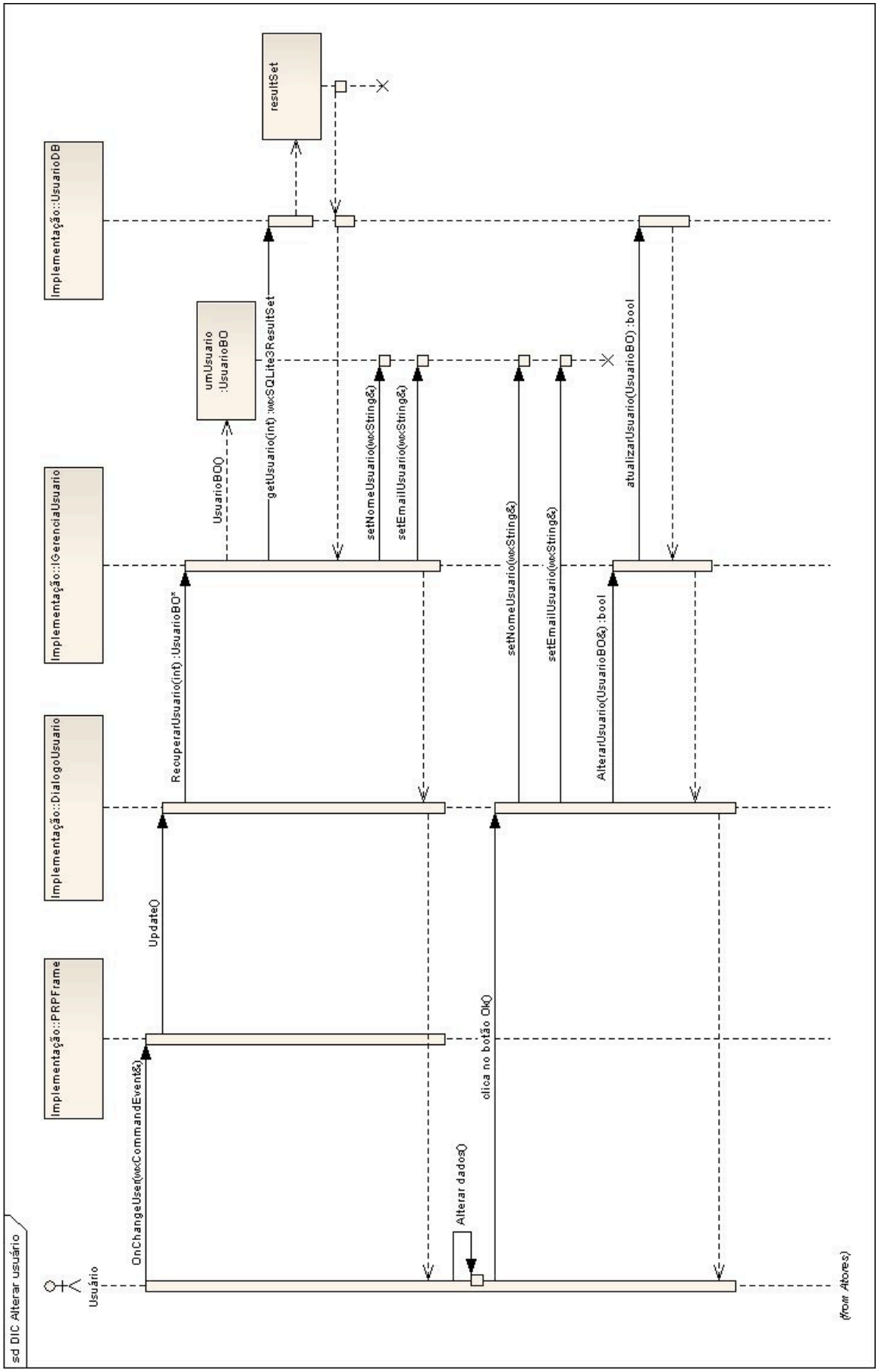

Figura 29 – Diagrama de seqüência do Alterar Usuário

4.3. Alterações e decisões

Durante o processo de desenvolvimento foram feitas algumas alterações de projeto em vista a adequar as novas tecnologia e versões de bibliotecas mais atuais. A mudança mais radical foi no quesito persistência, onde o SGBD foi modificado para um BD baseado em arquivo. Esta mudança foi realizada devido ao critério de instalação, visto que se fosse continuado a utilização de um SGBD o usuário final seria obrigado a instalar um sistema deste somente para dar suporte a persistência do PRP.

No caso foi escolhido o sistema de BD baseado em arquivo, SQLite, que vem a ser um banco de dados em arquivo e escrito em C++ contendo diversas bibliotecas pra acesso a dados. Esta modificação de projeto mostrou-se excelente devido ao fato que agora não é mais necessária a instalação de um SGBD para o correto funcionamento do PRP. Outra boa vantagem pela mudança foi à descoberta que este sistema de arquivos, o SQLite, é largamente utilizado em aplicações que funcionam em celulares e assistentes pessoais (PDA).

Outra decisão de projeto foi justamente a escolha por criar um sistema componentizado. Esta visão de construção de sistemas é bem moderna e trás diversos benefícios para a manutenção e evolução de um sistema construído sob este paradigma. Como o sistema é dividido em módulo, independentes entre si no funcionamento, toda alteração pode ser executada em um módulo sem que se afete o funcionamento do(s) outro(s), desde que se mantenham intactas as interfaces entre os módulos.

Esta abordagem de construção mostrou-se muito interessante no aspecto acadêmico, contudo durante o desenvolver do projeto provou sua real utilidade e facilidade. Durante o projeto foi necessários realizar-se alterações nas especificações do componente Financeiro, em vista a adaptar-se as mudanças nas legislações financeiras. Sendo assim foi provado que a teoria de construção de sistemas componentizados tem reais aplicações e já se encontra bem madura para sua larga utilização.

Até agora somente falamos das alterações e decisões que circundam a arquitetura do sistema em si, pois bem agora iremos falar sobre as decisões referentes aos requisitos do sistema. O PRP foi desenvolvido para ser um assistente de planejamento financeiro, tendo como seu carro chefe o controle de investimentos e um módulo de apoio a decisão, e é exatamente este módulo que foi alvo das maiores decisões do projeto.

Visto que a sua funcionalidade principal é a de criar uma extrapolação dos ganhos relacionados aos investimentos tivemos que adotar uma forma matemática de criar esta extrapolação. No início pensou-se em utilizar uma abordagem de inteligência artificial, contudo esta idéia foi logo descartada visto que somente sua complexidade iria sobrepor em muito o projeto do sistema. Outra abordagem foi utilizada que seria utilizando-se dados históricos criar funções que recriassem a curva destes dados no tempo.

Esta abordagem mostrou-se mais adequada a complexidade de um projeto desta ordem, sem termos em contrapartida uma perda significante de qualidade das previsões. Esta não perda de qualidade está pautada em uma já conhecida ferramenta de análise financeira a análise técnica, ou gráfica. Esta ferramenta muito utilizada no mercado de ações determina que os valores futuros dos papeis de uma empresa, estão fortemente atrelado a padrões que podem ser vistos na analise gráfica da evolução de tais ações ao longo do tempo.

Inicialmente quando foi pensado em utilizar-se da IA para fazer as previsões, o que foi pensado foi que diversos fatores externos, tais com, anos de eleições, crises internacionais, etc. poderiam e podem influenciar o andamento dos ganhos com investimentos. Contudo mais uma vez vamos ao mercado trazer uma solução, que é exatamente a análise técnica já mencionada anteriormente. Neste caso a outra premissa da ferramenta de análise técnica foi utilizada, que e, todos os efeitos que podem intervir no preço de uma ação já o fazem e sendo assim seu preço retrata a realidade que o mercado enxerga daquele papel.

4.4. Correções

Durante o projeto algumas correções foram feitas, mais precisamente na sua parte arquitetural. Inicialmente foi utilizada a biblioteca wxWidgets versão 2.6.3 e o SQLite 1.6, contudo ao longo o projeto diversas “features” novas foram acrescidas as bibliotecas e com isso após uma análise técnica foi decidido alterar as bibliotecas para as suas versões mais recentes, 2.8 e 1.9 respectivamente. Estas atualizações de bibliotecas proveram maior integração entre o sistema e seus dados, visto que a versão da biblioteca SQLite utilizada foi a wxSQLite.

Outras correções feitas foram na organização estrutural do projeto (número de classe, arquivos de apoio, etc.). A organização inicial mostrou-se confusa e pouco eficiente na

condução do projeto, assim foram feitas algumas alterações para deixar o projeto mais "limpo" e fácil de ser compreendido.

5. Conclusões e resultados

Diversas conclusões foram tiradas ao final do projeto, algumas técnicas, referentes às tecnologias/ferramentas utilizadas, outras de ordem gerencial, tais como, organização, cronograma, etc, e por fim conclusões referentes ao tema escolhido, estas baseadas mais nos resultados obtidos. Vamos as conclusões.

Na esfera técnica o que foi aprendido com o projeto é que a modelagem orientada a objetos mostrou-se muito mais fácil de ser administrada, alterada, corrigida, enfim ganhos sensíveis em relação a outras modelagens estruturadas. A orientação a objetos ajuda muito na representação da realidade, tomando para si os mesmos aspectos e características que vemos nos "objetos" do dia-a-dia.

Outra conclusão técnica refere-se a utilização de uma biblioteca de componentes gráficos "pré-construído". Isso se mostrou de extrema importância na velocidade do desenvolvimento, bem como na simplificação do desenvolvimento de aplicações com interface gráfica. Todos os erros e problemas encontrados no sistema final foram decorrentes da não familiaridade com as bibliotecas, seu funcionamento e suas exigências.

A biblioteca wxWidgets demonstra maturidade para ser utilizada em aplicações comerciais. Já o mesmo não pode ser falado da biblioteca de acesso aos dados, o wxSQLite. Esta biblioteca ainda encontra-se em um estágio menos evoluído que a wxWidgets, ainda com alguns erros e conflitos, contudo o conceito por trás da sua concepção, um SGBD em forma de arquivo, é extremamente válido e tem muito espaço no mercado para ser utilizado.

Quando ao *set* de ferramentas escolhidos no geral todas se mostraram excelentes no que se propõem a fazer, com destaque especial para a IDE de desenvolvimento, o Eclipse. Excelente ferramenta com um conjunto grande de *plug-ins* e uma enorme base de conhecimento distribuído em forma de fóruns, FAQs, How-tos, etc, na Internet. A utilização da linguagem C++ mostrou-e adequada ao propósito do projeto, uma aplicação *desktop stand-alone*.

As conclusões da área gerencial não foram muitas, contudo de forte impacto. Visto que o sistema foi 100% idealizado por mim, tendo seus requisitos expostos e escolhidos por

mim diante do propósito do projeto, não tivemos problemas de escopo com o projeto. Contudo é válido lembrar que, este é um dos problemas que mais atingem projetos na área de TI, com constantes mudanças de escopo, devido a refinamentos de conhecimento, vontade do cliente, mercado, etc.

Uma das conclusões gerenciais que tivemos neste projeto foi relacionada ao seu cronograma. Como este foi meu primeiro projeto, os erros referentes a prazo foram enormes, tanto nos prazos referentes à tecnologia, quanto no desenvolvimento. Os tempos estimados para estudos das tecnologias que seriam aplicadas no projeto foram subestimados, os tempos referentes à construção estouraram em muito o estimado. Isso ocorreu devido à falta de experiência de dados históricos de projetos anteriores nesta mesma linha.

Ainda na área gerencial vimos que a estrutura em que o projeto foi divido mostrou-se boa, bem modular, sem grandes variações no tamanho e complexidade dos pacotes de trabalho. O custo do projeto foi extremamente baixo devido ao fato de todo o *set* de ferramentas e bibliotecas serem gratuitos, tendo custos somente com alocação de pessoal. Acreditamos que o maior responsável pelos atrasos e desvios de cronograma do projeto tenham sido fatores ligados a qualidade do projeto. Para manter o sistema com qualidade foi gasto muito tempo na sua codificação para que esta fosse robusta e escalável, visto que o projeto possui outras fases a serem implementadas.

Em relação às conclusões do tema do projeto estas foram muitas. Inicialmente quero colocar que no meu ponto de vista o projeto atingiu o ponto a que se proponha, mesmo com algumas mudanças sendo necessárias. Foi constatado que no mercado não há aplicação completa que tenha um enfoque de gerenciamento de recursos pessoais em toda sua escala, indo desde simples contas a receber e pagar, até investimentos complexos. Algumas aplicações encontradas possuem *expertise* em um ponto e não em outro.

Do ponto de vista do cadastro de informações no sistema este se mostrou adequado ao uso diário para o controle das contas do usuário. Na parte do cadastro de investimentos tivemos algumas mudanças realizadas ao longo do projeto, visto um melhor entendimento do problema e possível solução. Como dito no item de correções, inicialmente o projeto teria a finalidade de realizar “previsões” de índice do mercado financeiro baseado em dados

histórico, contudo isto se mostrou inviável de ser executado dentro do nível de complexidade proposto no projeto.

Sendo assim optou-se por outra linha, a de benchmarking. Ou seja, o usuário cadastra seus investimentos iniciais, informa a movimentação através de lançamentos e diz qual será o índice de benchmarking deste investimento. Quando da análise financeira o sistema traça um comparativo entre o índice de mercado e o índice da conta, determinando se o investimento superou ou não o seu benchmarking naquele período.

Ao longo do projeto houve um aprimoramento muito grande do conhecimento do mercado financeiro e assim pudemos aplicar soluções mais customizadas para a nossa realidade. Coisas do tipo, impostos incidentes, taxas, índices de desempenho, até mesmo as peculiaridades de cada setor do mercado foram levadas em conta, seus jargões, dialetos, etc.

Mesmo o sistema tendo se mostrado de grande complexidade em sua primeira fase, concluímos que aplicativos nesta área ainda estão muito imaturos para virarem ícones no mercado e estarem instalados nos diversos computadores pessoais. Tive a visão que é uma área de desenvolvimento pouco explorada pelas fábricas de *software*, mas com bom potencial, tendo um ponto de atenção forte, as questões regulamentares do mercado em que o sistema ira estar inserido.

No Brasil temos diversas leis que até certo ponto são contraditórias e de confuso entendimento pela maioria dos brasileiros. Este ponto deve ser levado em consideração no desenvolvimento de um sistema para o mercado financeiro nacional. Alem é claro das questões de segurança das informações do usuário.

6. Revisão crítica e projetos futuros

Os resultados apresentados pelo sistema foram considerados satisfatórios diante da proposta feita no início do projeto. O sistema atendeu todos os requisitos funcionais levantados na fase de análise, sendo possível manter um padrão na interface gráfica, de modo a torná-lo mais amigável ao usuário, devido ao seu público-alvo.

O desempenho do sistema para uma massa de dados pequena foi considerado bom pelo desenvolvedor, uma vez que as respostas do sistema aos diversos estímulos do usuário não ficaram lentas, atingindo níveis bem aceitáveis inclusive pela complexidade de algumas tarefas. Não foi possível avaliar o desempenho para uma massa de dados grande. Outro fator importante que não pode ser testado é o limite de capacidade do banco de dados utilizado.

Devido à necessidade de manutenção de uma massa de dados remota que evolui bastante com o tempo e a sua dependência de uma fonte de dados única, que pode ser retirada de divulgação a qualquer hora (índice financeiros retirados de sites na Internet), fica muito difícil a comercialização do produto.

Para torná-lo comercializável, foram propostas algumas alterações, a seguir:

- Implementar o sistema utilizando uma solução de web-servers: O acesso aos dados necessários ao correto funcionamento do sistema seria feito diretamente entre a aplicação instalada e o fornecedor de tais informações através de tecnologia de web-servers;

- Integração com os bancos comerciais: Com o intuito de reduzir as entradas de informações por parte do usuário seriam feitas interfaces com dados dos bancos. Este ponto já foi estudado e mostrou-se viável visto que a maioria dos bancos possui a geração de extratos em formato específico e padronizado;

- Integração com BOVESPA: Existem alguns eventos financeiros relacionados a BOVESPA que impactam diretamente os resultados de uma operação como, por exemplo, o pagamento de dividendos e ou juros. Devido à limitação de sua fonte de dados, não foi possível contemplar esta funcionalidade, visto que estas informações

não ficam disponíveis de forma coesa e de fácil acesso. Neste ponto somente a entrada manual das informações é possível.

Outros fatores que poderiam dificultar a sua comercialização é o fato de o usuário ter que informar suas operações e questões relacionada a cultura de segurança da informação. Estas seriam barreiras culturais a serem vencidas, visto que o usuário deveria acreditar na segurança do sistema e a estar disposto a despender tempo na administração dessas informações.

7. Bibliografia

Pressman, Roger S. Engenharia de Software. Sexta Edição. McGraw Hill. 2006.

Sommerville, Ian. Engenharia de Software. Sexta Edição. Addison-Wesley. 2003.

Booch, G. Jacobson, I. Rumbaugh, J. UML - Guia do Usuário - 2ª Edição. Campus. 2006.

Elmasri, R. Navathe, S. B.Sistemas de Banco de Dados. Quarta Edição. Addison-Wesley. 2005.

Cormen, T. H. Leiserson, C.E. Rivest, R.L. Stein, C. Algoritmos – Teoria e Prática. Segunda Edição. Campus. 2002.

Ramalho, J. A. Microsoft SQL Server 2005 – Guia Prático. Campus. 2005.

Deitel, H. M. Deitel, P. J. C++ Como Programar. Terceira Edição. Bookman. 2001.

Julian, S. Kevin, H. Stefan, C. Cross-Platform GUI Programming with wxWidgets. Bruce Perens Open Source. 2005.

Blaha, M. Rumbaugh, J. Modelagem e Projetos Baseados em Objetos com UML 2. Campus. 2006.

Fowler, M. UML Essencial. Bookman. 2005.

Larman, C. Utilizando UML e Padrões. Bookman. 2007.

Gamma, E. Richard, H. Johnson, R. Vlissides, J. Padrões de Projeto. Bookman. 2000

Cheesman, J. Daniels, J. UML Components - A Simple Process for Specifying Component-Based Software. Addison-Wesley. 2000.

Resnick, P. RFC Network Working Group. Request for Comments: 2822. 2001.
Web link: ftp://ftp.rfc-editor.org/in-notes/rfc2822.txt , acessado dia 06/04/2008.

8. Referências técnicas

Biblioteca wxWidgets versão 2.8.10 – www.widgets.com

Biblioteca wxSQLite versão 1.9.5 – http://wxcode.sourceforge.net/components/wxsqlite3

Biblioteca wxChart versão 1.0.0 – http://wxcode.sourceforge.net/components/wxchart

IDE Eclipse Ganimede versão 3.4.2 – www.eclipse.org

Compilador MinGW versão – www.mingw.org

Java SDK versão 1.6 – www.sun.com

SQLIte Database Browser versão 1.3 – sqlitebrowser.sourceforge.net

Notação UML versão 2.1 – http://www.uml.org/

Notação UML Components versão 2.1;