# SFD – Sistema de Finanças Domésticas

**Felipe Pereira Gonçalves[1], Willian Ricardo Fialka Agner[2]**

[1],[2]Tecnologia em Analise Desenvolvimento de Sistemas
Faculdade Guairacá Guarapuava 2013

felipepg0@gmail.com, will.agner@hotmail.com

***Abstract****. The aim of this paper is to present the entire development of a software desktop personal financial control, with open source tools such as Java and Hibernate framework. The software is for users to have control over their financial condition, obtaining a control incoming and outgoing cash flow, also offering a simulated a future purchase to know if you can buy this product, the aim is to provide software that meets user needs and easily adaptable, taking into account the difficulty of finding such free systems and simplicity in use.*

***Resumo.*** *O objetivo desse artigo é apresentar todo o desenvolvimento de um software desktop de controle financeiro pessoal, com ferramentas open source como a linguagem Java e o framework Hibernate. O software é para que os usuários tenham controle sobre sua condição financeira, obtendo um controle de entrada e saídas de fluxo de caixa, também oferecendo um simulado de uma compra futura para saber se é possível comprar esse produto, o intuito é oferecer um software que atenda às necessidades do usuário de maneira fácil e de fácil adaptação, levando em consideração que a dificuldade de encontrar tais sistemas gratuitos e simplicidade no uso.*

## 1.0 Introdução

Para o desenvolvimento do trabalho, foi escolhido o ramo de finanças domesticas, há *softwares* nessa área porém a maioria são complexos demais e tem como foco a área empresarial, neste *software* foi concentrado no controle doméstico e de forma simples para que qualquer usuário possa utilizá-lo de forma direta e eficaz. Para o levantamento de requisitos do *software* a ser desenvolvido foi realizado entrevistas com um cliente que é formado em ciências contábeis e tem conhecimento profissional na área de finanças.

O sistema foi desenvolvido na linguagem Java com a ferramenta NetBeans 7.2 e o banco de dados com o MySQL, com o do sistema de gerenciamento de banco de dados MySQL WorkBench, também foi utilizado o *framework* de persistência de dados Hibernate. Os diagramas do *software* foram criados a partir do *software* Astah Community.

Fazer um bom controle financeiro demanda de uma certa paciência e tempo, porém essas coisas são muito raras de se obter ainda mais em nosso atual cotidiano. Foi analisado que o controle ideal é aquele que recebe todas as receitas e despesas, mesmo dos valores mais insignificativos que quando acumulamos geram um impacto significativo ao longo do ano. Porém poucas pessoas conseguem manter suas finanças tão organizadas, basta esquecer de lançarmos algumas despesas para perder o ímpeto do controle financeiro, de acordo com pesquisas feitas no site Minhas Economias (2013).

Com a utilização do *software* o usuário terá mais facilidade para o cadastro de suas despesas, pois será feito de forma simples e rápida, o usuário também terá acesso aos relatórios e poderá imprimi-los da forma que melhor lhe atender, escolhendo as datas das despesas assim podendo imprimir apenas em um certo intervalo de tempo sendo apenas o que lhe importa no momento.

## 2.0 Fundamentação Teórica

Na década de 90 o brasileiro estava acostumado com as elevadas taxas de inflação, onde os preços aumentavam e diminuíam quase que diariamente, com isso o brasileiro não criou hábitos de planejamento financeiro. Apenas com a implantação do Plano Real no Brasil, em 1994, iniciou-se um processo de estabilização econômica, possibilitando assim que as pessoas passassem a consumir mais, porém devido à falta de hábito de planejar as finanças pessoais, a população brasileira se endividou, assim as pessoas endividadas, sem dinheiro para cumprir com os seus compromissos passaram a ter diversos problemas, conforme pesquisas no site Brasil Escola (2013).

Após a estabilização financeira, as pessoas continuaram apresentando dificuldades em seus controles financeiros, pois as famílias, as escolas e o governo não ensinaram as crianças a se educarem financeiramente. Com a estabilização econômica também se tornou possível efetuar projeções quanto ao valor do dinheiro no futuro, assim aos poucos o planejamento financeiro familiar e pessoal passou a fazer parte da vida das pessoas brasileiras, de acordo com pesquisas realizadas no site Brasil Escola (2013).

É importante conhecer finanças pessoais, para podermos aumentar o nosso patrimônio financeiro, para tomarmos decisões eficazes, além de fazermos orçamento doméstico levando em conta até mesmo as despesas variáveis, pois temos que tomar decisões financeiras ao longo de toda nossa vida, a todo o momento, não somente investidores precisam tomar esse tipo de decisão e sim todo indivíduo, com base em pesquisas feitas no site Dinheiro Inteligente consultoria (2012).

O orçamento doméstico faz parte da educação financeira, ele nos ajuda a disciplinarmos e seguirmos um controle, com ele podemos acompanhar e verificar para onde está indo nosso salário e por que o dinheiro não dura até o final do mês. As pessoas têm necessidades limitadas, porém suas rendas são ilimitadas, por isso é necessário se fazer um orçamento doméstico, já que ele é a representação das receitas e despesas de todos os membros da casa, pois ele é útil para que você identifique quais são os itens que consomem a maior parte do seu dinheiro, e se isso é viável para a sua renda mensal, de acordo com pesquisas no site Brasil Escola (2013).

## 2.0 Ferramentas Utilizadas no Desenvolvimento

### 2.1.1 Java

De acordo com Deitel (2010) a *Sun Microsystems* financiou uma linguagem baseada em c++ em 1991 porém formalizou o Java apenas em 1995 quando a *web* já tinha ganhado muito popularidade, então o principal aspecto do Java que chamou a atenção da comunidade de negócios foi o interesse *web*. Assim tornou-se a linguagem preferida para implementar *software* da *web* ou que se comunicam por uma rede, como equipamentos de som estéreo e outro dispositivos domésticos que muitas vezes são

conectados em rede pela tecnologia Java. O Java é utilizado para aplicativos corporativos de grande porte e também para dispositivos moveis como celulares, *pagers* e PDAs.

A linguagem Java é OO (Orientada a Objetos) e segundo Barreiro e Sobral (2008) a orientação a objetos tem como objetivo modelar mundo real, assim melhorando a manutenção que devido a esse contexto será mais fácil de fazer, pois a orientação de objetos permite um desenvolvimento mais rápido, visto que o desenvolvimento ocorre através de blocos de códigos correspondentes aos objetos e acoplamentos, outra grande vantagem da OO é a reutilização que lhe permite utilizar objetos criados anteriormente e que podem ser facilmente acoplados na aplicação.

De acordo com Barreiro e sobral (2008) o Java tem como característica uma sintaxe muito parecida com C++, muitos julgam como uma versão simplificada do C++.

A portabilidade e segurança da linguagem são boas, isso deve-se ao JVM (Máquina Virtual Java) onde o aplicativo Java não entra em contado com o computador real, pois ele conhece apenas a JVM e é ela quem faz se necessário o serviço de entrada e saída, sistema de arquivos, memória, dessa forma o Java se torna seguro e portável para qualquer máquina ou sistema operacional.

### 2.1.2 Hibernate

Muitas aplicações principalmente corporativas manipulam grande quantidade de dados, e esses dados são armazenados em banco de dados relacionais, pois o modelo relacional é usado pela grande maioria dos sistemas gerenciadores de banco de dados, porém grande parte das aplicações corporativas são desenvolvidas com linguagens orientadas a objeto, como o modelo relacional e o modelo orientado a objetos são diferentes no modo de estruturar os dados, uma conversão deve haver toda vez que for ocorrer trafego de informações entre eles e essa conversão não é fácil pois os dois modelos são bem distintos, então no contexto da aplicação Java temos um *framework* de persistência chamado *Hibernate* que se encarrega de facilitar a conversão das informações entre a aplicação e o banco de dados, ela serve como um intermediário. Apostila K19 (2012)

O *hibernate* é um serviço de mapeamento objeto/relacional para o Java, é uma ferramenta de trabalho fácil e eficiente com informações de um banco de dados relacional na forma de objetos naturais do Java. O *Hibernate* 2.1 ganhou o 14º prêmio anual da Jolt da revista *Software Development* na categoria de bibliotecas, *framework* e componentes. James, Tim e Ryan (2009).

O *hibernate* tem um rotulo curioso “peso leve”, isto porque ele foi projeto para ser fácil de compreender e usar, os projetistas compreenderam as reais necessidades para sua execução, assim o deixando amplamente útil e profundo. James, Tim e Ryan, (2009).

O *Hibernate* realiza persistência no banco de dados através dos mapeamentos da classe Java, ou seja, o desenvolvedor deve criar referências do banco de dados, pois esse mapeamento permitirá que o *Hibernate* saiba qual tabela do banco de dados a classe Java representa, assim como os dados gerados devem ser persistidos, mas o *Hibernate* não faz apenas o mapeamento das classes Java para o Banco de Dados, mas também oferece consultas de dados e facilidades de recuperação que pode reduzir consideravelmente o tempo de desenvolvimento. O *Hibernate* tem como meta aliviar o desenvolvimento em 95% de dados comuns de persistência relacionadas as tarefas de

programação, pode ser que o *hibernate* não seja a melhor solução para as aplicações centradas em dados, no entanto o *hibernate* pode auxilia-lo a remover ou condensar o especifico código fornecedor de SQL. Manual de Referência *Hibernate* (2010).

### 2.1.3 MySQL

Um banco de dados é uma coleção de estrutura, ele pode ser desde a lista mais simples como uma galeria de imagens ou a uma grande quantidade de informações como uma rede coorporativa. Ele serve para adicionar, acessar e processar dados armazenados em um banco de dados de um computador é preciso de um sistema de gerenciamento de dados como um servidor MySQL, como os computadores são bons para em lidar com grandes quantia de dados o gerenciador de banco de dados serve como uma engrenagem central na computação. Manual de Referências do MySQL 4.1 (2010).

MySQL é um servidor robusto de banco de dados SQL (Linguagem Estruturada para Pesquisas), ele é multiusuário, multitarefa e muito rápido, pode ser usado em sistemas de produção com altíssima carga e missão crítica, também pode ser embutido em um programa de uso em massa. O programa MySQL tem duas licenças, onde os usuários podem escolher entre a gratuita e a comercial, onde sua versão gratuita é a mais popular. Manual de Referências do MySQL 4.1 (2010).

O MySQL é um sistema gerenciamento de dados relacional, ou seja, armazena dados em tabelas separadas em vez de colocar em apenas um local, assim proporcionando uma velocidade e flexibilidade muito melhor, pois ele foi desenvolvido para lidar com banco de dados grandes e de maneira rápida e confiável e apesar dele estar em constante desenvolvimento ele já oferece um ótimo conjunto de funções. Manual de Referências do MySQL 4.1 (2010).

Segundo Beighley e Morrison (2010) o banco de dados pode ser comparado a uma gaveta de arquivos bem arrumada e organizada, assim as informações ficam salvas de forma organizada facilitando a consulta exata quando solicitado. Os bancos de dados são gerenciados por um programa chamado servidor de dados, ou seja, o MySQL, pois ele se comunica com o servidor utilizando uma linguagem chamada SQL. No MySQL os dados são organizados em tabelas as quais armazenam as informações em forma de linha e coluna de dados relacionados, o deixando mais ágil e flexível.

### 2.1.4 MySQL WorkBench

O MySQL *WorkBench* é uma ferramenta visual de modelagem de dados com um vasto suporte as funcionalidades oferecidas pelo MySQL, possui como característica mais impactante a unificação de tarefas importantes como a analise, conhecimento e implementação de banco de dados em um único ambiente de desenvolvimento. MySQL *WorkBench Team* (2009).

O MySQL *WorkBench* é distribuído de duas formas a gratuita e a paga, a versão gratuita é denominada de *WorkBench OSS*, que não possui alguns recursos que a paga contém, porém não significa que não podemos utilizar em sua totalidade. MySQL *WorkBench Team* (2009).

A plataforma do MySQL *WorkBench* tem como uma das suas principais características, um ambiente de desenvolvimento agradável, de fácil uso,

proporcionando ao usuário melhores condições para a modelagem de banco de dados. MySQL *WorkBench Team* (2009).

### 2.1.5 JasperReport

*JasperReport* é um *software* gratuito que fornece relatórios e recursos de análise de dados, é um *software* de fácil integração com outros aplicativos, ele também permite facilmente criar novos relatórios, gerencia de forma eficiente e segura, tem boa interação com os relatórios. *JasperReports Server User Guide* 4.7 (2012).

*JasperReports* tem uma aspecto importante que é o *layout* do relatórios definido em um arquivo XML, esse XML possui todas as informações de formatação do relatório, ele é capaz de exportar relatórios para diversos formatos diferentes como PDF, XML, XLS, etc. Aceita diversas formas de entrada de dados com o XML e o CVS, tem conexão com o banco de dados, uma sessão com *Hibernate*, uma coleção de objetos em memória, etc. Também permite o uso de gráficos, diagramas e código de barras. *JasperReports Server User Guide* 4.7 (2012).

### 2.1.6 NetBeans IDE 7.2

NetBeans IDE versão 7.2 (*Integrated Development Environment* – Ambiente Integrado para Desenvolvimento) e uma ferramenta gratuita para desenvolvimento *desktop*, *mobile* e *web*, com suporte para várias linguagem inclusive o Java, segundo o próprio site (NetBeans 2013) o NetBeans oferece o melhor suporte às tecnologias Java, pois ele oferece um suporte abrangente e recente para as tecnologias Java.

A plataforma NetBeans é um *framework* genérico para aplicações *Swing*, ele serve como um "canalizador", pois antes o desenvolvedor tinha que escrever o código conectando ações de itens no menu, barra de ferramentas e atalhos de teclado, agora o NetBeans lhe fornece isso de uma forma mais prática e simples, você não precisa mais codificar essas e outras características básicas, e o melhor que a plataforma não acrescenta grande sobrecarga para sua aplicação e o usuário ganha economiza tempo, ele também lhe proporciona uma arquitetura de aplicativo confiável e flexível, assim podendo ser utilizados até mesmo em *softwares* de grandes portes, de acordo com pesquisas no próprio site da (NetBeans 2013).

Segundo Oliveira (2008) o NetBeans também possui suporte para *frameworks* de teste, servidores web, monitoramento de aplicação, entre outras ferramentas, tem vantagens como gerar automaticamente partes do código assim poupando tempo do desenvolvedor, também para testes pode-se utilizar o próprio console da plataforma, pois esta apresenta um maior números de caracteres por linha assim facilitando a visualização dos resultados.

### 2.1.7 XAMMP

De acordo com pesquisas feitas no site oficial (XAMPP, *Apache Friends* 2013) o XAMPP tem um pacote de componentes para rodar uma aplicação que necessite de um servidor *web*, ele possui algumas vantagens como a fácil instalação, e também possui o servidor necessário além do MySQL, Apache, Perly e PHP, ele também roda em vários sistemas operacionais como *Linux, Windows, e Mac OS*.

O Xammp foi usado principalmente para monitoração do banco de dados, devido a facilidade de utiliza-lo e a experiência com o mesmo durante o período de aulas.

### 2.1.8 Astah Community

*Astah Community* é uma ferramenta de *design* de sistema que suporta UML e disponibilizam uma versão do *software* gratuito, que já é capaz de criar vários diagramas, assim fazendo a modelagem dos diagramas de classe, sequencia, caso de uso, entre vários outros, *Astah Manual reference* (2012).

## 3.0 Metodologia de Desenvolvimento

### 3.1 MVC

Segundo Martin Fowler (2003) o MVC (Figura 1) é um dos padrões mais usados e citados, ele começou como um *framework* desenvolvido por Trygve Reenskaug para a plataforma *smaltalk* no final dos anos 70, desde então ele tem exercido um papel de influências na maioria dos *frameworks* e considera três camadas:
***Model* (modelo):** É um objeto não visual que contém todos os dados e comportamentos da interface do usuário, na sua forma mais pura o modelo é um objeto dentro de um modelo de domínio.
***View* (visão):** Representa a interface para o usuário com base nos dados do modelo, a view diz respeito apenas a apresentação de informação, pois quaisquer alterações nessas informações são manipuladas pelo controller.
***Controller* (controlador):** É responsável por controlar toda a aplicação, pois é ele quem controla a entrada do usuário manipula o *model* e faz com que a *view* seja atualizada corretamente.

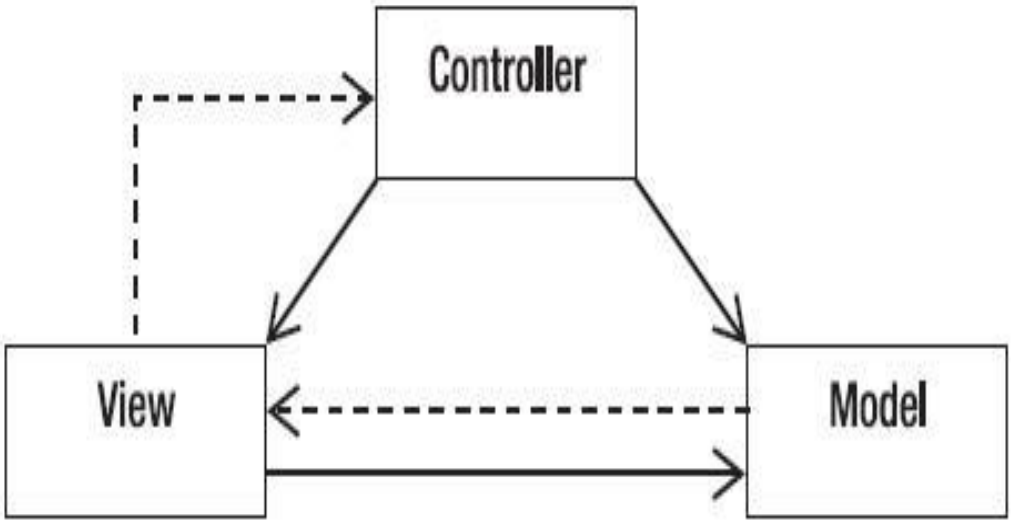



**Figura 1. *Model View Controller***
**Fonte: http://www.devmedia.com.br/introducao-ao-padrao-mvc/29308**

### 3.2 UML

De acordo com Guedes (2005) a UML (Linguagem de Modelagem Unificada) é uma linguagem visual para modelar *softwares* em um paradigma de orientação a objetos, e foi muito bem aceita pelas indústrias de engenharia de *software*, deve

ficar bem claro que a UML não é uma linguagem de programação e sim uma linguagem de modelagem onde são criados vários diagramas onde os engenheiros podem modelar todo o software e assim definir melhor suas características, requisitos, comportamento, etc.

Guedes (2005) afirma que os diagramas tem o objetivo de fornecer várias visões do sistema a ser modelado, assim cada diagrama completa o outro, pois cada diagrama tem um papel especifico sendo uns mais abrangentes e outro mais específicos, a vantagem é que muitas vezes um diagrama corrige os outros e assim evita erros e atrasos no desenvolvimento.

## 4.0 Desenvolvimento

Para o desenvolvimento desse software foi utilizado o ciclo de vida cascata (Figura 2) de acordo com Pressman (2011) o ciclo de vida cascata sugere uma abordagem sequencial e sistemática para o desenvolvimento do *software*, começando pelo levantamento de requisitos através do usuário e vai avançando sequencialmente pelas fases de planejamento, modelagem, construção, emprego e culminado no suporte continuo do *software* concluído.

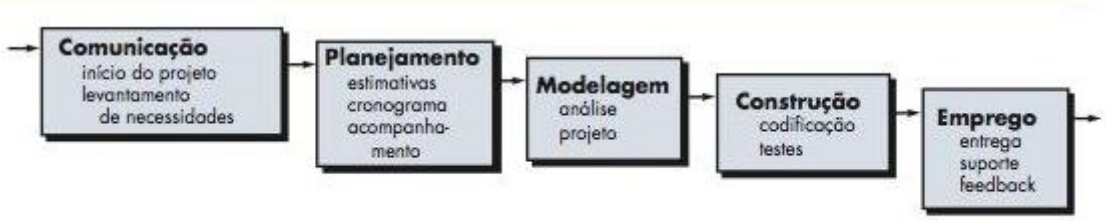



**Figura 2. Cascata**

**Fonte: http://www.galeote.com.br/blog/2012/06/modelos-prescritivos-para-o-desenvolvimento-de-software/**

O desenvolvimento do sistema iniciou-se com entrevistas com o cliente que tem experiência no ramo de negócio e operacional, com base nas entrevistas foi realizado o levantamento de requisitos, então foi dado início a criação dos casos de uso (apêndice 1), diagramas de classe (apêndice 2) e diagrama de sequência (apêndice 3)através do *software Astah Community*, pois a ferramenta é de fácil interação com o usuário e também tem licença gratuita.

Através da ferramenta *WorkBench* foi feito a modelagem do banco de dados onde foram criadas e vinculadas as tabelas necessárias, tudo isso com base no diagrama de classes já criado.

Após estar pronto o banco de dados e com o *Hibernate* devidamente configurado e com as *Annotations* prontas iniciou-se o desenvolvimento das telas do sistema, e com as persistências no banco através do *Hibernate*. O sistema contém três pacotes de códigos Java principais (*Bean, View, DAO*).

Onde o pacote bean contém as classes com seus atributos e as *Annotations* do *hibernate* que modelaram o banco de dados, nessa fase pode-se notar a agilidade em criar e modelar o banco de dados com hibernate. No pacote *View* desenvolvemos a interface do sistema, a interface é o meio de comunicação entre o usuário e o sistema, inicialmente foi desenvolvido uma tela de login para o sistema com verificação de login e senha para proteger as informações do usuário.

Após o login foi desenvolvido a tela principal, essa *interface* é fixa e maximizada para o usuário, é através dessa *interface* que o usuário poderá ter acesso as demais funcionalidades do sistema como cadastro de usuários, gastos e receitas, realizar prospecção e consultar relatório dos mesmo, ainda nessa tela o usuário tem algumas consultas básicas como as receitas e os gastos do mês atual.

A *interface* de cadastro de usuário o usuário poderá cadastrar novos usuários para o sistema, onde para ele cadastrar basta colocar o nome e a senha que deve ser digitada duas vezes para confirma-la, o usuário também poderá deletar outros usuários através dessa interface basta digitar o nome e senha do usuário que deseja excluir, então o sistema verificará o login se estiver correto as informações ele excluirá o mesmo, o usuário também tem a possibilidade de alterar a senha.

A *interface* de cadastro de contas o usuário deve apenas cadastra-las com o nome, já na *interface* de lançamento de gastos o usuário seleciona uma conta já cadastrada e assim pode lançar os valores, datas de vencimento e até mesmo quantidade de parcelas, foi optado a fazer dessa maneira para uma melhor organização do sistema e assim o usuário pode apenas escolher a conta já cadastrada e lançar vários gastos para a mesma.

Na *interface* da renda mensal o usuário insere suas receitas também com a possibilidade de escolher datas e como uma breve descrição para saber do que se trata a receita, pode-se selecionar como uma receita fixa assim todo mês ela será adicionada a sua renda, um exemplo disso é para um usuário que recebe uma receita fixa por mês como um salário fixo.

Depois foi desenvolvido a *interface* de prospecção, onde sua função é calcular a possibilidade do usuário comprar determinado produto futuramente, ou seja, lhe informando se é possível ou não fazer essa compra, pois essa interface faz um cálculo com base no valor do produto e em quantas parcelas pretende pagar se for parcelar, assim ele faz uma média de seus futuros gastos e receitas, assim obtendo uma base dos seus saldos futuros e dessa forma verificando a sua condição financeira para obter esse novo produto.

Por fim foi feito a *interface* de relatórios e detalhes onde o usuário tem uma visão personalizada das suas receitas e gastos, logo que a interface é iniciada ela carrega as tabelas com todas suas receitas e gastos, porem o usuário tem a possibilidade de controlar isso pelas datas, onde escolhe a data inicial e a data final para trazer os detalhes e sempre trazendo o saldo final que consiste em suas receitas subtraídas pelos gastos.

Em todas interfaces foi adicionado botões com imagens referente a funcionalidade do botão, dessa forma deixando-o com um modelo visual mais elegante e também facilitando na identificação dos botões.

Com o *bean* e a *view* pronta demos início ao desenvolvimento do pacote DAO, onde ocorre o CRUD (*Create Read Update Delete* – Criar Ler Atualizar Excluir), através dessas classes faz toda a parte de persistência no banco de dados, foi utilizado uma classe chamada AbstractDAO que funciona como uma classe genérica para fazer as funções de salvar, atualizar e excluir assim diminuindo a quantia de códigos no sistema e facilitando o desenvolvimento e futuras manutenções.

Cada classe *Bean* tem uma classe DAO essas classes tem como principal intuito fazer as consultas no banco de dados, para fazer as consultas utilizamos a *Criteira* de acordo com pesquisas feitas no site (IBM 2012) o *Criteria Query API* permite a construção de *query expression* em Java, o Criteria permite de forma programática construir objetos query através da *interface org.hibernate.Criteria*, que define os possíveis métodos para se trabalhar com o Criteria, sendo mais simples que as consultas através do HQL e SQL.

Com a ferramenta jasperReport foi criado os relatórios do sistema, onde o usuário poderá verificar e imprimir os resultados de suas pesquisas.

## 5.0 Resultados

Com o desenvolvimento do projeto pode-se observar a importância do controle de finanças pessoais, pois ele sendo devidamente utilizado o usuário tem um controle fiel das suas finanças, assim podem se planejar melhor para o futuro, como por exemplo para fazer as prospecções o sistema acaba por administrar suas finanças auxiliando assim a comprar o produto que ele deseja.

Outro ponto importante notado no projeto é que há possibilidade de desenvolver um *software* de qualidade com ferramentas *open source*, e também o vasto material sobre as mesmas ferramentas em *sites* e fóruns, dessa forma servindo de aprendizagem para qualquer pessoa com uma base de aprendizagem que queira começar a desenvolver sistemas.

O *framework hibernate* ajudou no desenvolvimento do trabalho com uma economia de tempo e código, pois suas funcionalidades de persistência em banco de dados foi de grande vantagem, assim descartando outros códigos e funcionalidades, tornando uma estrutura mais simples e funcional.

Sobre as ferramentas utilizadas para o desenvolvimento desse projeto, todas atenderam às expectativas pois possibilitaram a redução no tempo de execução das tarefas, de uma forma simples, funcional e sobretudo profissional.

## 6.0 Considerações Finais

Foi alcançado o objetivo proposto no início do projeto, pois o *software* é capaz de cadastrar as movimentações financeiras do usuário e grava-las em um banco de dados, assim apresentando quando solicitado, os resultados financeiros auxiliam na tomada de futuras decisões, principalmente com o uso da prospecção auxiliando, o usuário poderá visualizar futuras aquisições, assim podendo se preparar de uma forma mais estratégica e dessa forma não correndo o risco de endividamento.

## 7.0 Projetos Futuros

Como trabalho futuro pretendemos expandir as plataformas do *software*, como implementar o *software* na *web*, dessa forma o usuário com conexão à *internet* poderá acessar as suas finanças em qualquer lugar e de qualquer aparelho com acesso à *internet*.

Como o sistema foi desenvolvido em Java, pretendemos implementa-lo em *android*, assim o deixando portátil e também podendo ser acessado em qualquer local, porém sem a necessidade de *internet*.

## Referências


Apostila de Treinamento K19 - Persistência JPA e Hibernate (2012) disponível em http://www.k19.com.br/downloads/apostilas/java/k19-k21-persistencia-com-jpa2-e-hibernate acessado em 01/10/2013.

Astah (2013) disponivel em http://astah.net/tutorials acessado em 25/09/2013.

Brasil Escola - Inflação brasileira disponível em http://www.brasilescola.com acessado em 10/10/2013.

Criteria Hibernate disponível em https://www.ibm.com/developerworks/community/blogs/fd26864d-cb41-49cf-b719-d89c6b072893/entry/criteria_hibernate_na_pr_C3_A1tica?lang=en acessado em 10/10/2013.

Daniela Barreiro Claro e João Bosco Mangueira Sobral (2008), Programação em Java.

Deitel (2010) Java como Programar, 8ª Edição.

Galvin King, Christian Bauer, Max R. Andersen, Emmanuel Bernard e Steve Ebersole Documentação de Referência Hibernate disponível em http://docs.jboss.org/hibernate/orm/3.5/reference/pt-BR/pdf/hibernate_reference.pdf acessado em 22/09/2013.

Gilleanes T.A. Guedes (2005) UML2, 2ª Edição, NOVATEC.

Ireports Team (2012), JasperReports Server User Guide disponivel em http://community.jaspersoft.com/sites/default/files/docs/jasperreports-server-user-guide.pdf acessado em 12/09/2013.

James Elliot, Tim O´Brien e Ryan Fowler (2009), Dominando Hibernate. Altas Books.

Lynn Beighley e Michael Morrison (2010), Use a Cabeça! PHP e MySQL, Alta Books.

Qual a importância do orçamento doméstico? Disponível em http://economia.culturamix.com/dinheiro-2/qual-a-importancia-do-orcamento-domestico acessado em 10/10/2013.

MMySQL Team (2006), Manual de Referência do MySQL 4.1 disponivel em http://downloads.mysql.com/docs/refman-4.1-pt.pdf acessado em 25/09/2013

MySQL WorkBench Team (2009) disponível em http://downloads.mysql.com/docs/workbench-en.pdf acessado em 15/10/2013.

Martim Fowler (2003) Arquitetura de Aplicação Corporativas, BOOKMAN.

Minhas Economias – Controle financeiro em 15 minutos (2013) disponível em http://www.minhaseconomias.com.br/blog/educacao-financeira/como-controlar-planejar-suas-financas-de-forma-simples acessado em 10/10/2013.

NetBeans (2013) disponivel em https://netbeans.org/ acessado em 05/10/2013.

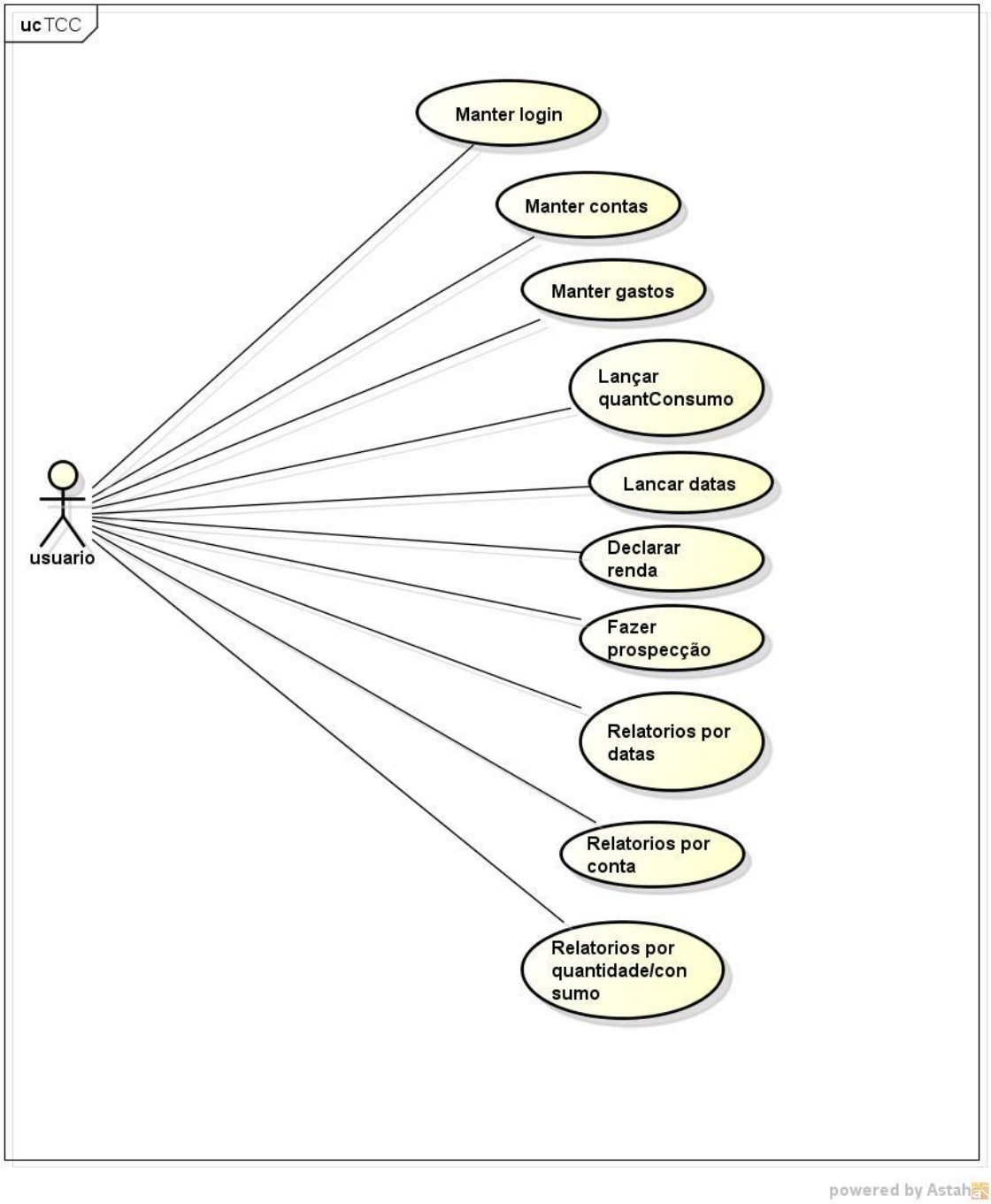
Oliveira, Cicero David Leite – Desenvolvimento de componentes Web Services disponibilizados por meio de Web Services. Instituto Tecnológico de Aeronáutica (2008) – disponível em http://www.bibl.ita.br/xivencita/COMP01.pdf acessado em 28/10/2013.

Pressman, S. Roger (2011) Engenharia de Software, 7ª Edição, AMGH.

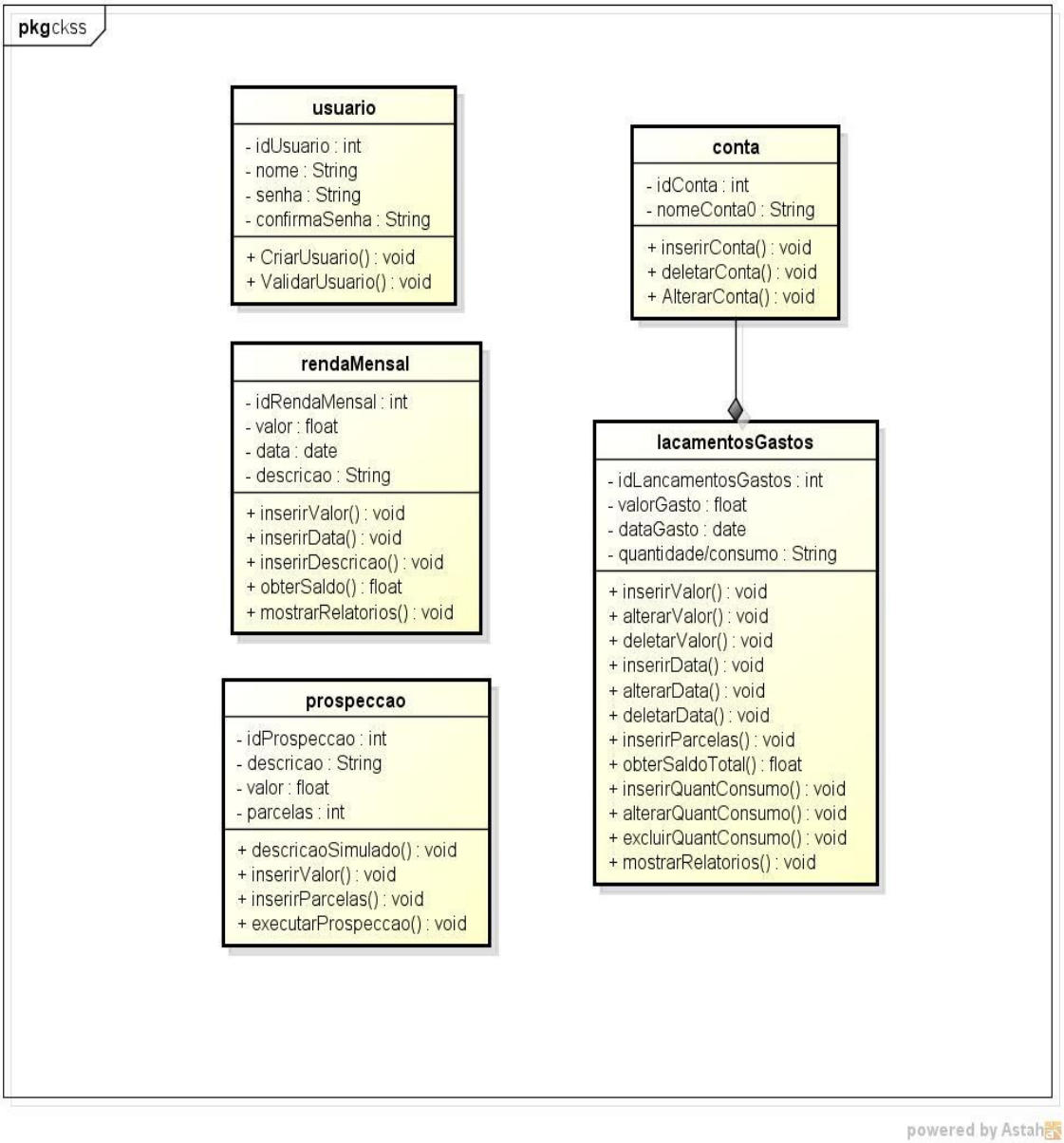
Planejamento financeiro porque é importante? Disponível em http://www.dinheirointeligente.com.br/website/artigo.asp?cod=1741&idi=1&id=16877 acessado em 10/10/2013.

XAMMP Friends (2013) disponivel em http://www.apachefriends.org/pt_br/xampp.html acessado em 13/10/2013.

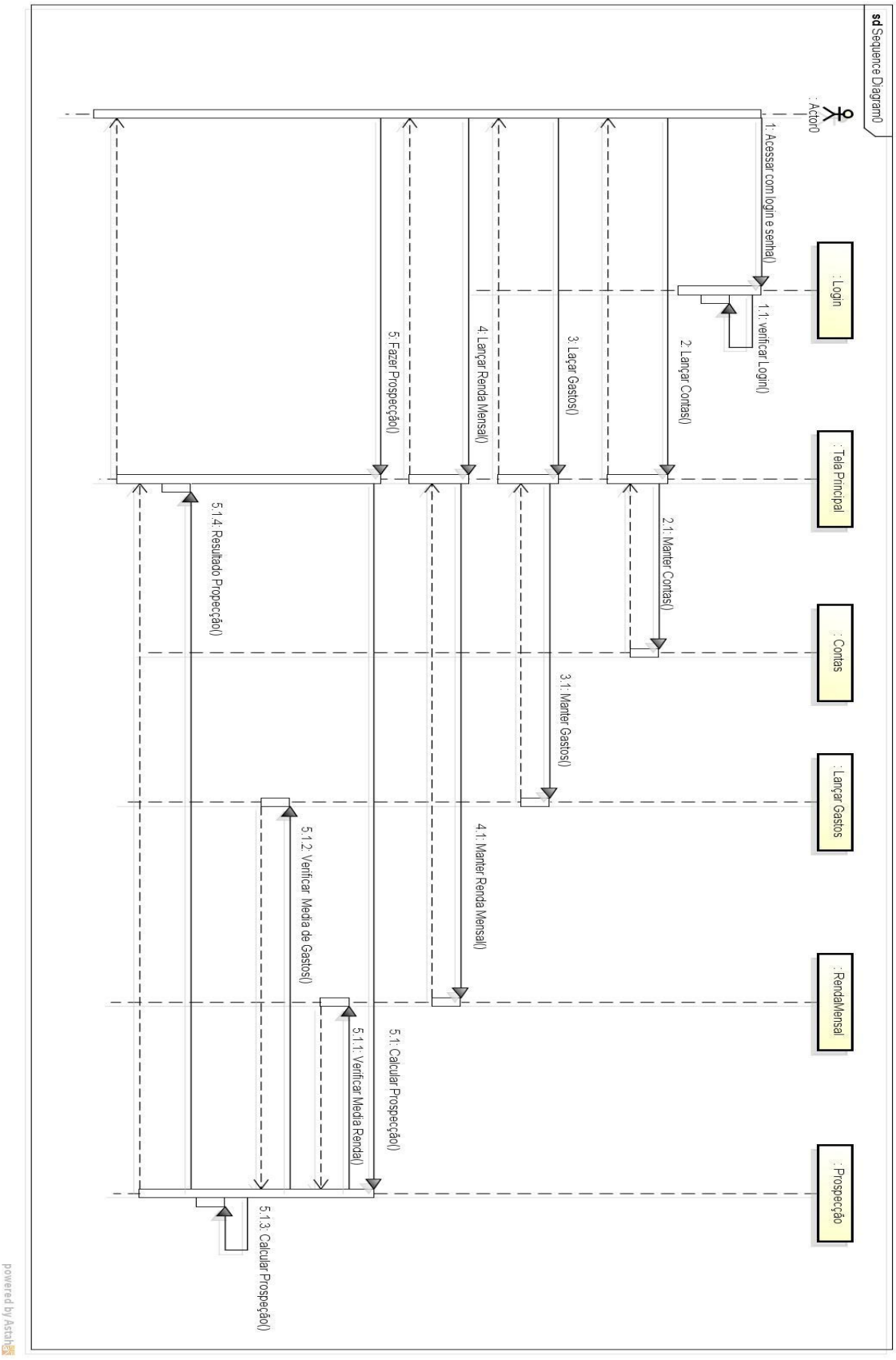
## Apendices



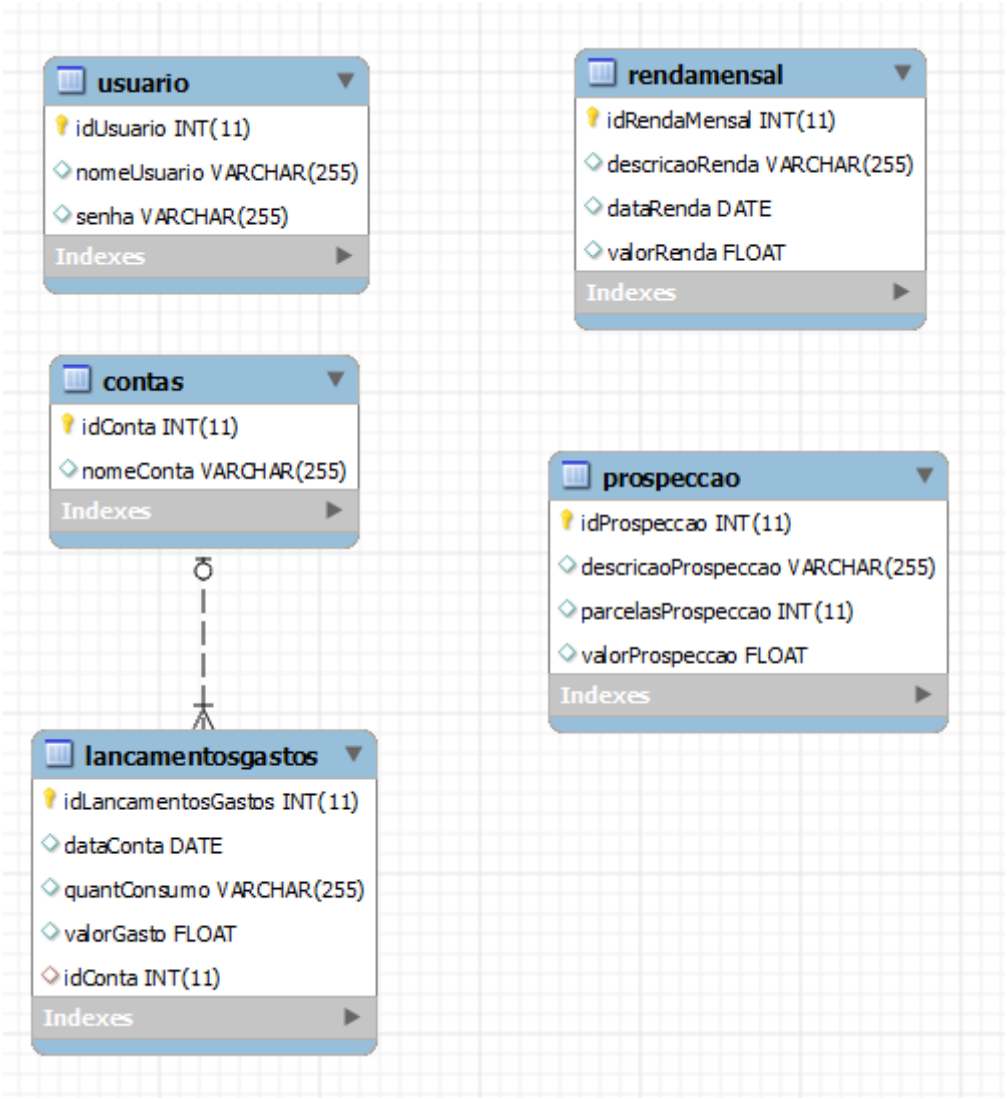

**Apendice 1 – Caso de Uso**




**Apendice 2 – Diagrama de Classe**




**Apendice 3 – Diagrama de Sequência**




**Apendice 4 – Modelagem Lógica**