

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
UNIDADE ACADÊMICA DE SISTEMAS E COMPUTAÇÃO
GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

PYFINANCIAL

DAVID CANDEIA
DIEGO D. DE FREITAS
EVERTON L. G. ALVES
FELIPE L. COUTINHO

PROF. DR. HYGGO ALMEIDA
(ORIENTADOR)

CAMPINA GRANDE, PARAÍBA, BRASIL
© DAVID CANDEIA, DIEGO D. DE FREITAS, EVERTON L. G. ALVES E FELIPE L. COUTINHO,
2009

Conteúdo

1	Introdução	1
1.1	Contexto do Projeto	3
1.1.1	Pessoas e Processos	3
1.1.2	Ambiente de Execução	3
1.1.3	Aplicações Correlatas	4
2	Fundamentação Teórica	5
2.1	Conceitos	5
2.1.1	Considerações iniciais	5
2.1.2	Valor Presente (VP)	6
2.1.3	Valor Presente Líquido (VPL)	6
2.1.4	Valor Futuro (VF)	7
2.1.5	Taxa de juros (I) e Taxa interna de retorno (TIR)	7
2.1.6	Número de Períodos (n)	8
2.1.7	Prestações (pmt)	8
2.1.8	Sistema de amortização (SA)	8
3	Processo de Desenvolvimento	9
4	Análise de Requisitos	15
4.1	Modelo Conceitual	15
4.1.1	Model	16
4.1.2	GUI	16
4.1.3	Controller	16
4.1.4	ModelListener	17

4.1.5	Reg	17
4.1.6	FinancialLibrary	17
4.2	Requisitos	17
5	Arquitetura	21
5.1	Descrição da Arquitetura	21
5.1.1	Camada de Apresentação	22
5.1.2	Camada de Aplicação	23
5.1.3	Camada de Persistência	23
5.1.4	<i>Packaging</i>	24
5.1.5	Pontos Adicionais	24
6	Validação	25
7	Métricas	27
8	Conclusões e Trabalhos Futuros	30
A	Glossário	32
B	Fórmulas	37
C	Novas Funcionalidades	42
C.1	Adicionando uma função	42
C.1.1	Considerações Iniciais	42
C.1.2	Exemplo de Adição de Função	44
D	Tabela de Erros	46
E	Diagrama de Classes	49
F	Telas da Aplicação	52
F.0.3	Tela Principal	52
F.0.4	Tela de Exceção	53
F.0.5	Telas de Amortização	53

F.0.6	Telas de Recall	53
-------	---------------------------	----

Capítulo 1

Introdução

Muito tem se discutido hoje em dia a respeito da necessidade de um melhor gerenciamento de finanças pessoais por parte dos membros da sociedade. Cada vez mais as pessoas percebem como é importante administrar bem seus recursos financeiros de modo a possibilitar o alcance de várias metas como, por exemplo, aquisição de imóveis, veículos, móveis de melhor qualidade ou até mesmo a aposentadoria. Outro ponto bastante importante é que munido dos conhecimentos básicos, a pessoa torna-se um consumidor muito mais responsável e apto a lidar com o mercado, bem como a realizar investimentos. Porém, em especial na sociedade brasileira, fala-se muito a respeito de possibilidades de investimento sem que antes a população receba uma educação financeira adequada [Educação 2009].

O tópico de matemática financeira é comumente lecionado nas escolas e análises a respeito de como esses cursos estão sendo conduzidos podem ser encontradas em [Azevedo]. Porém, ao analisarmos o comportamento das pessoas no dia a dia do comércio são poucos os que têm uma noção clara dos princípios financeiros e que consequentemente os usam rotineiramente. Logo, o uso de calculadoras financeiras facilitam o aprendizado e o exercício dos conceitos e pode auxiliar nos cálculos do cotidiano. Além desse ponto, os especialistas confirmam que a matemática financeira não é mais praticada atualmente sem o uso de tais dispositivos [dos Santos 2001], em especial devido à complexidade de certos cálculos que tomariam muito tempo para serem desenvolvidos manualmente.

Nesse contexto, a HP-12C [HP] se coloca como uma calculadora líder de mercado há vários anos, inclusive com vários modelos que foram desenvolvidos ao longo desse tempo, e muitos são os cursos e livros que ensinam Matemática Financeira com o apoio da mesma.

É fato que para profissionais da área como Administradores, Economistas, dentre outros, a mesma se apresenta como uma boa calculadora e bastante confiável. Entretanto, ao analisar usuários que usam alguns dos conceitos financeiros mas não são especialistas é comum observar-se uma certa rejeição a este tipo de calculadora, devido à complexidade para realizar certas análises como, por exemplo, a de um plano de amortização. Um produto que seja mais prático facilitaria o aprendizado e uso desses conceitos.

Observando um pouco a história recente dos dispositivos móveis percebe-se o grande crescimento do uso dos mesmos no cotidiano das pessoas [G1 2009]. Conjuntamente com esse aumento de uso/venda, surgiu também um forte mercado para o desenvolvimento de aplicativos para dispositivos móveis e hoje são inúmeros os softwares que podem ser adquiridos para as mais variadas plataformas, tanto através de compra nos sites oficiais de fornecedores, de desenvolvedores autônomos ou ainda através da pirataria. Esses aplicativos tipicamente perpassam o mundo da multimídia, como jogos, players, etc., ao da organização de tarefas, com calendários e agendas, dentre outros.

Entretanto, não é comum vermos aplicativos relacionados ao mundo financeiro. Recentemente um grande avanço nessa área foi o lançamento de um emulador da HP-12C para o Iphone da Apple [Mac 2009]. Logo, observa-se aqui uma boa oportunidade de alavancar um pouco mais esse nicho. Para tal, firmou-se como objetivo do projeto o desenvolvimento de dois artefatos: i) uma biblioteca de funções, escritas em Python, que implementem as principais funções financeiras existentes na HP-12C. Com tal resultado obtém-se um auxílio para o desenvolvimento de novos aplicativos na área financeira, dado que, de acordo com as pesquisas realizadas, percebemos a inexistência de pacotes de software que forneçam certas operações básicas que comumente estão presentes nas várias aplicações financeiras; e ii) uma calculadora financeira para os dispositivos da Nokia, mais especificadamente para o N800, a qual foi designada de *PyFinancial*, que terá como objetivo implementar de forma mais amigável, intuitiva e completa os requisitos essenciais para cálculos financeiros.

Do ponto de vista funcional, a calculadora *PyFinancial* é um software com propósitos financeiros que irá oferecer ao usuário o cálculo das várias funções financeiras básicas existentes na HP-12C, como análises de fluxos de caixa, bem como cálculo de planos de amortizações. A solução proposta visa oferecer um nível de confiança similar ao da HP-12C, com resultados bem próximos aos que são encontrados na mesma, e, tirando proveito das

facilidades de interação que o dispositivo nos provê, oferecer uma maior praticidade no cálculo das várias funções financeiras ali presentes. A interface *touch-screen* que o dispositivo oferece é uma dessas facilidades que torna o uso da calculadora mais prático.

1.1 Contexto do Projeto

1.1.1 Pessoas e Processos

Conforme relatado acima, o perfil principal de usuários a serem alcançados com a aplicação são aqueles que não possuem conhecimentos aprofundados em matemática financeira e que consequentemente necessitam de uma interface mais amigável para a realização dos cálculos.

Entretanto, não pode-se descartar, devido à facilidade de interação buscada, aqueles usuários que desejam aprender a respeito de matemática financeira. Esses podem encontrar na calculadora uma boa oportunidade para de fato realizar seu aprendizado, e esse grupo de usuários vem tornando-se crescente, conforme já relatado anteriormente. Além destes, pode-se conseguir atrair também aquelas pessoas que fazem uso desses cálculos cotidianamente, mas que acham complicado a execução de certas operações nas calculadoras existentes no mercado. Para tanto, buscou-se opiniões com profissionais e alunos da área financeira de modo a aglutinar idéias de bons formatos de interação entre o usuário e a aplicação.

1.1.2 Ambiente de Execução

Para a implantação da aplicação faz-se necessário que o usuário possua:

- Dispositivo Internet Tablet N800 da Nokia [Nokia a]. Atualizações do código para o dispositivo N810 estão no planejamento de evolução da calculadora.
- Sistema operacional Maemo Diablo (4.1.x) [Maemo.org a].
- Ambiente gráfico QT4 [Maemo.org c] e PYQT4 [Maemo.org d]
- Python 2.5 [python.org].

É importante destacar que não foram realizados testes com nenhuma versão posterior as que foram acima relatadas. O motivo principal para a não realização desses testes é a

instabilidade de algumas das novas versões.

1.1.3 Aplicações Correlatas

Nesta seção será relatado um pouco a respeito de algumas aplicações similares a calculadora *PyFinancial*:

- *Mobile Financial Calculator V1.0* [FarsightSoft.com]: Essa calculadora se assemelha a aplicação proposta na facilidade de uso, utilizando-se de vários menus, e oferece um conjunto variado de funções. Porém, destina-se a dispositivos um pouco mais antigos, por exemplo dispositivos Série 60 [Nokia b], o que não lhe permite um alto grau de interação, como o proposto pela *PyFinancial* através da interface *touch-screen* do N800.
- *HpCalc-Iphone* [Fors]: É um emulador da calculadora HP-12C disponibilizado para o dispositivo Iphone da Apple. Embora forneça um alto grau de interação através da interface *touch-screen*, a mesma, por ser um emulador, é uma cópia fiel da HP-12C, o que implica na permanência das dificuldades de uso para usuários menos habituados com os conceitos financeiros.
- *Web HP-12C emulator* [Pfützenreuter]: É um emulador da HP-12C oferecido na Internet que, de maneira similar ao emulador anterior, possui as mesmas dificuldades de uso já relatadas, mas possui um problema mais grave que diz respeito à acurácia de seus resultados. Comparou-se resultados oferecidos pela mesma com resultados da HP-12C original e foram percebidas distorções consideráveis em certo conjunto de cálculos, por exemplo, nos cálculos do pagamento e número de períodos.
- *Finance Calculator Version 4.2* [Lutus b]: É uma calculadora financeira oferecida na Internet desenvolvida em JavaScript e que possui um alto grau de facilidade em seu uso, bem como uma alta acurácia nos resultados apresentados em comparação com a HP-12C original, inclusive com documentação das fórmulas usadas. A grande fraqueza da mesma é o fato de disponibilizar apenas os cálculos dos valores financeiros básicos como valor presente, valor futuro, pagamento, número de períodos e taxa de juros.

Capítulo 2

Fundamentação Teórica

Matemática financeira é um ramo da Matemática que se utiliza de uma série de conceitos matemáticos para realizar a análise de dados financeiros em geral. Para a implementação da *PyFinancial* foram considerados os principais conceitos financeiros existentes na HP-12C, conforme presente no Apêndice A.

É importante ressaltar que durante a pesquisa realizada no intuito de coletar as fórmulas financeiras relacionadas aos conceitos implementados, percebeu-se que várias são as fórmulas existentes e divulgadas na Internet, porém muitas delas não apresentam resultados similares aos da HP-12C. Com isso, à medida que são apresentados os conceitos é interessante observar as fórmulas que estão implementadas e que se encontram no Apêndice B.

Abaixo tem-se uma breve explicação dos principais conceitos financeiros utilizados na *PyFinancial*.

2.1 Conceitos

2.1.1 Considerações iniciais

Aplicações financeiras partem de um capital inicial e este é aplicado a uma certa taxa de juros, podendo ser esta aplicação configurada por uma série simples ou complexa, de modo a gerar, com o passar do tempo, um capital final que implique em alguma remuneração ou perda. Esses juros são aplicados ao capital de acordo com a capitalização fixada para o investimento. Como exemplos temos investimentos em poupança, empréstimos, etc.

Uma forma simples de ter-se uma idéia de como funciona uma dessas movimentações é observando o fluxo de caixa da mesma através de um diagrama de tempo. Um exemplo desse diagrama pode ser visto na Figura 2.1. Nele temos representado uma entrada no tempo inicial e a realização de dois pagamentos em dois instantes distintos de tempo.

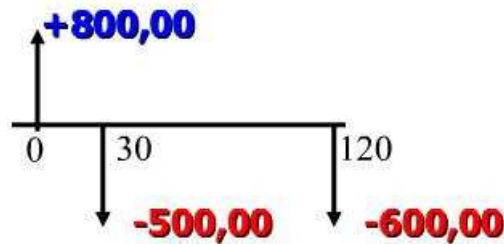


Figura 2.1: *Diagrama de Fluxos de Caixa.*

2.1.2 Valor Presente (VP)

Aplicações têm início com uma quantia a partir de um marco inicial. Essa quantia corresponde à quanto vale o investimento inicialmente, sem análises do futuro. A esse valor, que pode ser um valor gasto ou recebido, dá-se o nome de Valor Presente ou valor atual. Um exemplo seria uma pessoa que aplica R\$ 500,00 na poupança. Esse valor seria o valor presente de sua aplicação. Assim, valor presente, capital inicial e valor atual têm o mesmo significado.

O valor presente de uma dada prestação pode ser entendido como sendo o valor no instante zero de uma certa prestação futura descontada dos juros.

2.1.3 Valor Presente Líquido (VPL)

Uma vez que o valor presente está definido, bem como as prestações, juros da movimentação e número de períodos durante os quais a aplicação estará vigente, uma análise bastante comum é verificar qual o valor da aplicação atualmente. Ou seja, para cada prestação descontam-se os juros existentes, sejam estes simples ou compostos, considerando a quantidade de períodos até que a prestação fosse paga e obtém-se o valor da prestação na data-pólo

inicial. Em seguida somam-se os valores presentes de cada prestação ao valor presente e obtém-se o VPL.

Um VPL positivo é um indicativo de que a aplicação possui um retorno positivo para o aplicante e que pode ser considerada. Já um VPL negativo indica que a aplicação não deve ser considerada. [Wikipedia c]

Uma consideração importante a ser realizada é que esse valor presente líquido pode ser calculado de duas formas diferentes: uma com pagamentos postecipados e outra com pagamentos antecipados.

2.1.4 Valor Futuro (VF)

Considerando o valor presente, os juros, as prestações e um dado período de vigência, o valor futuro de uma dada aplicação é o valor obtido ao final da mesma. Ou seja, o valor futuro é o resultado da incidência dos juros e das prestações sobre o valor presente. Esse valor também é conhecido como capital final.

É importante destacar que, da mesma forma que o VPL, o valor futuro é influenciado pelo fato de estar-se trabalhando com pagamentos antecipados ou postecipados. Logo, dependendo do tipo de pagamento em vigência têm-se diferentes valores futuros.

2.1.5 Taxa de juros (I) e Taxa interna de retorno (TIR)

Os juros, ou taxa de juros, de uma dada aplicação é o percentual que será aplicado sobre um montante de modo a gerar a remuneração da aplicação. Essa remuneração pode ser feita através de um sistema de juros compostos ou ainda de um sistema de juros simples.

A taxa interna de retorno, por sua vez, é a taxa de juros que iguala o valor presente das entradas ao longo da aplicação ao valor presente das saídas da aplicação. Ou seja, podemos entendê-la como o valor da taxa de juros que torna o valor presente líquido do investimento igual a zero, a aplicação não fez diferença.

Se a taxa de juros considerada para a aplicação for menor que a TIR então é um indicativo de que a aplicação é atrativa, se for maior a aplicação não deve ser considerada e sendo igual implica que a aplicação não fará diferença[Wikipedia b].

Por exemplo, suponha que uma empresa invista R\$ 30.000.000,00 para obter fluxos fu-

turos de R\$ 11.000.000,00, R\$ 12.100.000,00 e R\$ 13.310.000 ao longo de três anos. A taxa interna de retorno do investimento é de 10% ao ano. Assim, se a taxa de juros do mercado fosse de 6% teria-se um VPL de R\$ 2.321.648,00, ao passo que se a taxa de juros fosse 15% teria-se um VPL de R\$ -2.533.903,00. Portanto, conclui-se que quanto maior for a TIR, maior será o número de possibilidades do investimento ser lucrativo.

2.1.6 Número de Períodos (n)

O número de períodos indica quantas vezes a taxa de juros será aplicada em uma certa aplicação de modo a gerar remuneração. Além disso, o número de períodos também informa os locais no tempo onde pode haver entradas ou saídas no fluxo.

2.1.7 Prestações (pmt)

As prestações indicam entradas ou saídas de capital em um certo fluxo de caixa. Tipicamente nos cálculos considera-se uma prestação negativa para representar uma saída de capital e uma prestação positiva para representar uma entrada. Por exemplo, pode-se pensar em uma empresa que recebe um montante de R\$ 60.000,00 a cada ano e gasta um total de R\$ 20.000,00, tem-se que o *pmt* a ser considerado para a aplicação da mesma é R\$ 40.000,00.

2.1.8 Sistema de amortização (SA)

Um sistema de amortização indica como um determinado financiamento é pago ao longo do tempo. Para isso, definem-se o valor das prestações que inclui tanto o montante a ser abatido do saldo devedor, bem como os juros a serem pagos. Assim, pode-se entender como amortização a diminuição do saldo devedor, ou dívida, que pode ser realizada em etapas ou de uma única vez.

Dentre os sistemas de amortização mais famosos é comum citar o Sistema de Amortização Francês (SAF), ou *Tabela Price*, e o Sistema de Amortização Constante (SAC). Atualmente o SAC é bastante utilizado no Sistema de Financiamento Habitacional, em especial pela Caixa Econômica Federal, e o SAF é amplamente utilizado pelos bancos em seus sistemas de Crédito Direto ao Consumidor (CDC), bem como nas vendas a prazo divulgadas pelas grandes redes de varejo [Meirelles 2009].

Capítulo 3

Processo de Desenvolvimento

Conforme as boas práticas que a Engenharia de Software prega, para que um projeto de software obtenha êxito quanto ao cumprimento de seus requisitos, é necessário que este seja desenvolvido de maneira sistemática e organizada. Por isso, processos de desenvolvimento de software são criados para auxiliar esta sistematização, consequentemente agregando valor ao trabalho dos desenvolvedores e ao produto final. Atualmente, organizações regulamentadoras como a ISO [ISO], utilizam como parâmetro de classificação de qualidade a verificação de quais processos de software são empregados nas instituições.

Conhecendo a grande importância que um processo de desenvolvimento exerce sobre a qualidade do produto final, buscou-se um processo, dentre os existentes na literatura especializada, que mais se adequasse ao contexto do projeto. Dentre os processos existentes optou-se pela escolha do XP1 [XP1], os motivos para deste ante os demais processos existentes foram os seguintes: i) trata-se de um processo bastante simplificado e indica a produção de um conjunto limitado de artefatos, logo, não traria maior *overhead* para a equipe de desenvolvedores; ii) trata-se de um processo desenvolvido por um conjunto de alunos e professores da Universidade Federal de Campina (UFCG). Tal fato trouxe a segurança que as possíveis dificuldades que aparecessem poderiam ser facilmente dissipadas pois a equipe de desenvolvimento estaria em contato direto com os autores do processo; iii) este processo tem obtido grandes sucessos no contexto acadêmico; e iv) o processo em si agrupa um conjunto de práticas já bastante utilizadas por empresas e segue as diretrizes principais de um bom processo de desenvolvimento conforme indica a Engenharia de Software.

É importante comentar que o processo XP1 foi escolhido como base para realização do

nosso projeto, porém como acontece na maioria das empresas que adotam um processo, foi necessário realizar algumas adaptações do mesmo para melhor refletir o contexto ao qual o projeto, ora descrito, estava inserido.

XP1 define um conjunto de papéis que os membros da equipe deverão assumir no decorrer do desenvolvimento, bem como quais responsabilidades cada pessoa que assumir tal papel terá. Os papéis, bem como as suas respectivas responsabilidades, estão listados a seguir:

Papel: Cliente

- Descrever a funcionalidade desejada.
- Descrever os requisitos não funcionais do software.
- Definir o plano de release de software.
- Descrever os testes de aceitação para cada *User Story*.
- Escolher *User Stories* para cada iteração.

Papel: Desenvolvedor

- Ajudar a levantar *User Stories* e requisitos não funcionais junto ao cliente.
- Elaborar um projeto arquitetural.
- Estimar o tempo de desenvolvimento de *User Stories* e tarefas.
- Elaborar o esquema lógico dos dados.
- Escrever o código das tarefas e Testes de Unidade.
- Executar atividades de integração e *Test Review*.
- Implementar a automação de Testes de Aceitação.

Papel: Gerente de Projeto

- Conduzir as atividades de planejamento.

- Alocar *reviewers* de testes.
- Avaliar riscos e lidar com os riscos descobertos.
- Manter o progresso do projeto.

Tipo de Atividade	Atividade	Quando realizar
Planejamento	Escrita de User Stories	No início do projeto, e sempre que o cliente pensar em novas stories
	Levantamento de requisitos não funcionais	No início do projeto, em paralelo com o levantamento de User Stories
	Planejamento de releases	Logo depois que o projeto arquitetural estiver pronto
	Planejamento de iteração	Iterações individuais são planejadas em detalhe imediatamente antes do início da iteração e não antecipadamente. Há uma iteração a cada 2 semanas.
Projeto	Projeto arquitetural	Imediatamente após o levantamento de User Stories e requisitos não funcionais
	Projeto do esquema lógico dos dados	No início de um release, planeja-se o modelo de dados para todas as User Stories do release. Mudanças poderão ocorrer durante iterações.
	Refatoramento constante	Ao longo das iterações
Testes	Elaboração de Testes de Aceitação	O cliente deve se antecipar e ter testes de aceitação prontos o quanto antes. Os testes de aceitação de uma User Story devem necessariamente estar prontos antes de iniciar qualquer tarefa relacionada ao User Story;
	Elaboração de Testes de Unidade	À medida que se codifica. Sugere-se codificar os testes de unidade <i>antes</i> de escrever o código. É uma boa forma de fazer o design de baixo nível do software.
	Realização de Test Review	Antes de fazer o check-in no final de uma tarefa.
Integração	Iniciar controle de versão	Na criação de qualquer documento, arquivo, etc. que necessite estar sob controle de versão.
	Realizar check-out	No início de uma tarefa. Também é feito no final de uma tarefa para realizar a integração (operação de update).
	Realizar check-in	No final de uma tarefa, depois de passar por um Test Review.
Gerência	Gerenciar riscos	Todo dia, em papos informais no corredor. Também é feito na reunião semanal de acompanhamento.
	Manter o progresso	A coleta de informação para o Big Chart deve ser feita pelo menos semanalmente, mais frequentemente se a coleta for automática.
	Publicação	A publicação de resultados de acompanhamento (tabela de riscos, Big Chart) deve ser feita pelo menos 1 vez por semana <ul style="list-style-type: none"> • Big Chart: antes da reunião de acompanhamento • Tabela de riscos: depois da reunião de acompanhamento • Plano de releases: sempre que houver mudança

Figura 3.1: Tabela que descreve quando as atividades de XP1 devem ser realizadas.

Instanciando os papéis identificados anteriormente (cliente, desenvolvedor e gerente) no contexto do projeto obtém-se a seguinte classificação: i) o professor Dr. Hyggo O. de Almeida assumiu o papel de cliente; ii) quanto ao papel de gerente, cada um dos integrantes da equipe assumiu a chefia do grupo por um determinado tempo durante o desenvolvimento (aproximadamente um mês cada); e iii) por tratar-se de um número reduzido de pessoal para execução do trabalho, durante o decorrer do projeto todos integrantes foram desenvolvedores/testadores.

Quanto à alocação de atividades (descritas na Tabela 3.1 retirada de [XP1]), houve sempre a preocupação em dividi-las igualmente entre os membros da equipe. Para tal,

reuniões de acompanhamento foram realizadas semanalmente. Nessas reuniões os membros da equipe procuravam alocar, segundo as habilidades de cada indivíduo, as atividades do modo mais adequado possível. Não havendo acordo, ficava a cargo do gerente da vez impor sua decisão final.

Adaptando as atividades elencadas pelo XP1 ao contexto do projeto percebe-se a seguinte configuração:

- A atividade de planejamento consumiu bastante tempo ainda na primeira parte da disciplina de Projeto I. Isto deve-se ao fato que esta etapa envolveu uma ampla pesquisa na literatura especializada (e discussões com o cliente) para verificação de quais requisitos, funcionais e não funcionais, deveriam estar presentes no produto final. Com os requisitos já identificados e com as *User Stories* escritas, o próximo passo foi realizar o planejamento das iterações e *releases*. É válido salientar que apesar de todo esse planejamento ter sido realizado durante as primeiras etapa do processo, a cada semana, durante as reuniões de acompanhamento, uma revisão sobre esses artefatos era realizada para garantir que o que se havia planejado realmente estava de acordo com a realidade do problema e da equipe. Portanto, alterações pontuais nos artefatos de planejamento foram realizadas durante todo o processo de desenvolvimento.
- A atividade de projeto foi iniciada imediatamente após a conclusão da etapa de planejamento. De posse das *User Stories* e dos requisitos do sistema a equipe em conjunto montou o projeto arquitetural do sistema. O projeto arquitetural decidido foi simplificado, baseado principalmente nas ideias de *Model-View-Controller* (MVC), porém atendeu as necessidades ali presentes. Quanto ao esquema lógico, por tratar-se de um projeto que não envolve grandes manipulações de dados, a equipe decidiu por não adotar um banco de dados, e apenas trabalhar com arquivos XML, quando necessário.
- Dada a importância que a atividade de testes tem para um bom produto final, a equipe buscou construir suites de teste de qualidade para que estas pudessem localizar rapidamente possíveis deficiências do código. Para tal, logo que as *User Stories* foram escritas os testes de aceitação das mesmas já começaram a ser escritos em sequência. E, seguindo o que indica a metodologia de testes *Test-Driven Development* (TDD) [Beck 2003], os testes de unidade foram sempre desenvolvidos antes que a codifi-

cação fosse realizada. É importante destacar que como o projeto foi proposto por um cliente que não era da área financeira, os testes de aceitação foram escritos pelos desenvolvedores junto ao mesmo a partir de materiais de exercícios financeiros encontrados na Internet, incluindo os manuais da HP-12C [HP 2009a] [HP 2009d] [HP 2009b] [HP 2009c] e o material de matemática financeira do professor Adail Marcos Lima Da Silva [Silva 2009], sempre comparando os resultados obtidos com os da HP-12C real. Outro ponto importante é que existiu sempre a preocupação de que pessoas diferentes escrevessem e revisassem os testes, procurando, assim, garantir uma maior credibilidade.

- A atividade de integração foi uma preocupação constante da equipe desde a fase de planejamento. Para que não ocorressem divergências ou inconsistências de conteúdo, surgiu a preocupação de criar um controle de versões tanto para a biblioteca financeira quanto para a calculadora *Pyfinancial*, para tal fez-se uso das já muito conhecidas ferramentas de controle de versões SVN [Tigris.org] e Garage [Maemo.org b]. O SVN foi utilizado para controle apenas da biblioteca financeira, dado que ela é um software livre e *open-source*, de modo a ganhar visibilidade para a mesma entre desenvolvedores Python, ao passo que o Garage foi utilizado para a calculadora financeira procurando dar-lhe visibilidade junto a plataforma Maemo. Com isso, manteve-se um controle sistematizado do código, bem como dos artefatos de planejamento.
- Quanto à realização de *Reviews*, sempre foi realizado um tempo após a escrita dos testes e do código funcional para a revisão da qualidade dos testes escritos (verificando novamente os resultados com a calculadora, observando coerência e cobertura dos mesmos, etc.), bem como para realizar algum refatoramento que se mostrasse necessário de maneira imediata. Embora tenha-se procurado organizar o design de maneira flexível, percebeu-se no decorrer do projeto que alguns pontos poderiam sofrer um refatoramento maior. Essa necessidade possivelmente está relacionada com o fato de não ter sido alocado desde o início um tempo maior para *Code* e *Design Review*.
- Por fim, como já dito anteriormente, a atividade de gerência foi revezada entre os membros da equipe. Cada um atuou como gerente por um determinado período de tempo

durante a realização do projeto. Ao gerente coube entender as mudanças discutidas durante as reuniões de acompanhamento e refleti-las tanto nos artefatos de planejamento (*User Stories*, requisitos, projeto arquitetural etc), quanto nos artefatos de gerência (*Big-Chart*, a tabela de riscos e o plano de *releases* etc). Essa atividade foi realizada sempre em paralelo com o desenvolvimento da aplicação, aproximadamente uma vez a cada semana.

Para garantir que as atividades fossem realizadas da forma mais adequada possível uma infra-estrutura foi montada a fim de melhorar a organização e consequentemente a qualidade do código produzido. Dentre os elementos presentes nessa infra-estrutura podemos destacar:

- **Eclipse** [Foundation]. IDE de desenvolvimento.
- **PyDev** [Aptana]. *Plug-in* para desenvolvimento em Python.
- **PyUnit** [Purcell]. *Framework* para desenvolvimento de testes de unidade.
- **PyEasyAccept** [Sauvé et al.]. *Framework* para desenvolvimento de testes de aceitação.
- **SVN e Garage SVN** [Tigris.org][Maemo.org b]. Ferramentas para controle de versões.

Capítulo 4

Análise de Requisitos

4.1 Modelo Conceitual

Na Figura 4.1 apresenta-se o modelo conceitual da calculadora financeira, objetivo central desse projeto. É possível identificar algumas entidades centrais desse sistema, são elas:

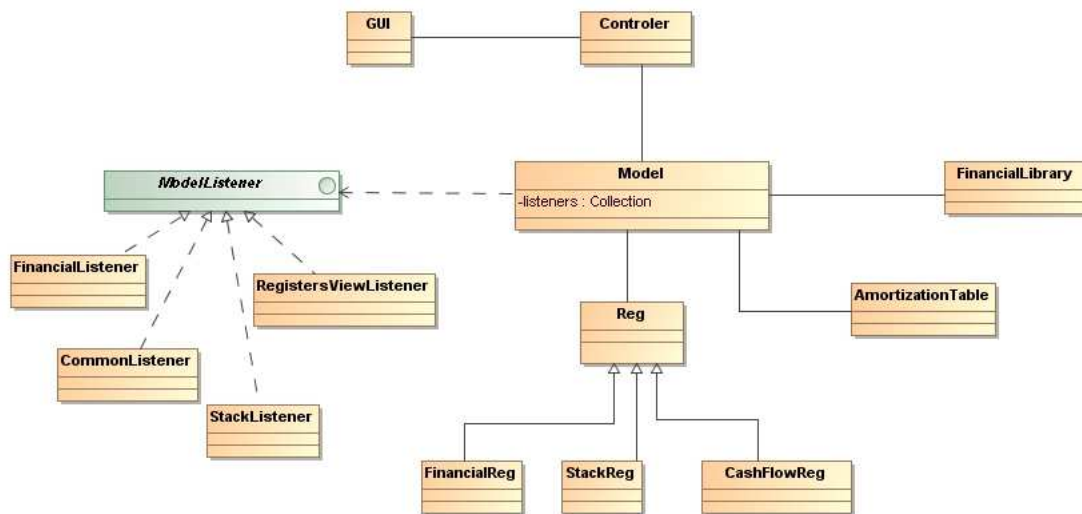


Figura 4.1: *Modelo Conceitual*.

- Model
- GUI
- Controller

- `ModelListener`
- `Reg`
- `FinancialLibrary`

Um maior detalhamento de como está organizado o código da aplicação pode ser encontrado no Apêndice C, onde são expostas informações a respeito de como alterar funcionalidades na calculadora, e no Apêndice E.

4.1.1 Model

Módulo central da aplicação que é responsável por fazer a troca de mensagens entre as demais entidades do sistema às quais está conectado, bem como realizar toda a manipulação de registradores (pilha, registradores financeiros e demais registradores comuns) e funções de modo a efetivar as funcionalidades requisitadas pelo usuário. Também é de sua responsabilidade receber e preparar os dados da aplicação para posterior manipulação, dados esses captados pela interface. Por fim, esta entidade também é responsável por persistir os registradores quando necessário, bem como todo o estado interno da calculadora previamente configurada pelo usuário.

4.1.2 GUI

Esta é a entidade que está em contato direto com o usuário da aplicação. Logo, este é o módulo que recebe os dados externos e os transmite para entidades mais internas, assim como recebe os resultados das operações realizadas pela calculadora e os transmite ao cliente de uma maneira mais amigável. Essa interface também é responsável por apresentar os dados segundo preferências do usuário previamente selecionadas, por exemplo, a formatação de um número segundo um determinado limite de casas decimais.

4.1.3 Controller

Esta entidade é o controlador do sistema, responsável por desacoplar a interface da camada de negócios do sistema. Recebendo as requisições feitas pelo cliente da aplicação, através

da interface (*GUI*), e identificando as operações correspondentes no *Model* que se responsabilizarão por realizar as atividades correspondentes.

4.1.4 **ModelListener**

Entidade abstrata da qual outras estenderão. Estas são responsáveis por capturar qualquer mudança importante ocorrida no *Model* que seja de interesse do usuário. Assim, essas alterações serão refletidas na interface. Como mudanças encaixam-se situações como requisições de realização de operações matemáticas/financeiras, manipulações de registradores, etc.

4.1.5 **Reg**

Entidade que representa o papel de um registrador. Um registrador é um importante elemento usado para a realização das atividades objetivo da calculadora. Neles armazenam-se os valores a serem utilizados nas operações, bem como os resultados das operações requisitadas. Esta entidade é especializada por outras adicionando as particularidades pertinentes aos diferentes tipos de registradores.

4.1.6 **FinancialLibrary**

Entidade que representa a biblioteca das funções financeiras. Todo e qualquer cálculo financeiro deve chamar a operação correspondente na biblioteca garantindo uma melhor organização e, conseqüentemente, a separação de interesses.

Essa é a entidade que poderá vir a ser utilizada por programadores Python, incluindo os membros da equipe, para desenvolvimento das mais variadas aplicações que necessitem da realização de funções financeiras.

4.2 **Requisitos**

Durante a etapa de planejamento, foram realizadas várias reuniões entre os membros da equipe, o cliente e o professor Adail de modo a, a partir do que foi levantado através da

pesquisa sobre as aplicações correlatas, definir um conjunto de requisitos funcionais e não funcionais. O conjunto de requisitos definidos podem ser observados abaixo:

- **Requisitos Funcionais**

- O usuário deverá ser capaz de realizar as operações matemáticas/financeiras seguindo a Notação Polonesa Reversa [Wikipedia a].
- O usuário deve ser capaz de especificar o número de casas decimais apresentadas, bem como requisitar a apresentação dos valores em notação científica.
- O usuário deve ser capaz de adicionar/limpar valores em todos os 30 registradores existentes na calculadora.
- O usuário deve ser capaz de realizar operações matemáticas básicas: soma (+), subtração (−), divisão (/) e multiplicação (*). Deve também ser capaz de realizar operações mais avançadas: exponenciação (e^x), quadrado de um número (x^2), logaritmo natural (\ln), potência de números (y^x), raiz quadrada ($x^{1/2}$), percentual do total ($\%T$), variação percentual ($\Delta\%$) e percentual de um número ($\%$).
- O usuário deverá ser capaz de realizar cálculos que descubram os principais valores financeiros: número de períodos (n), taxa de juros (i), valor presente (VP), valor da parcela (PMT) e valor futuro (VF). Esses cálculos podem envolver ou não múltiplos fluxos de caixa.
- O usuário deve ser capaz de realizar análises de investimentos mais apuradas através do cálculo do valor presente líquido (VPL), bem como da taxa interna de retorno (TIR).
- O usuário deve ser capaz de calcular a amortização de suas dívidas no sistema de amortização francês (SAF), bem como no sistema de amortização constante (SAC). Apresentando-se uma tabela de amortização ao final.
- O usuário deve ser capaz de realizar a conversão entre taxas de juros anuais para taxas de juros mensais tanto no sistema de juros simples, como no sistema de juros compostos. Além disso, deve ser capaz de converter um número de períodos anuais em períodos mensais.

- O usuário deve ser capaz de rotacionar os valores dos registradores da pilha de cima para baixo.
- Ao finalizar o uso, os dados da tela e dos registradores devem ser armazenados em algum tipo de persistência.

- **Requisitos Não-Funcionais**

- **Interface.** Possuir interface similar a da calculadora HP12-C, com melhoramentos que possam facilitar sua utilização.
- **Usabilidade.** O nível de dificuldade encontrado para uso da calculadora deve ser o mesmo encontrado quando faz-se uso das calculadoras tradicionais.
- **Volume de Utilização.** A calculadora será monousuário, porém devendo ser robusta o suficiente para ser utilizada durante um longo intervalo de tempo ininterruptamente sem apresentar problemas.
- **Hardware e software alvo.** A calculadora deverá funcionar no Internet Tablet N800 da Nokia, no sistema operacional Maemo Diablo (4.1.x)
- **Qualidade.** Com relação a precisão, deseja-se que os valores calculados na calculadora não difiram em mais de 0.001 unidades dos valores calculados na HP12-C original.
- **Expressividade nas mensagens.** Entradas equivocadas do usuário deverão apresentar mensagens de erros que sejam mais intuitivas do que as apresentadas pela calculadora original (e.g. "Erro 6") [Apêndice D].
- **Desempenho.** O tempo de resposta dos resultados não deve ultrapassar o tempo que a calculadora HP12-C leva para executar as mesmas operações (considerando que só o programa da calculadora esteja em execução).
- **Segurança.** Os arquivos que conterão os valores dos registradores recém utilizados pelo usuário deverão ficar protegidos de alteração externa.
- **Compatibilidade.** É desejável que a aplicação possa ser portada para versões mais novas do dispositivo alvo, como o N810, bem como para novas versões do Maemo.

- **Internacionalização.** A calculadora deve ser desenvolvida para tornar possível a internacionalização.
- **Packaging.** A distribuição do aplicativo deve ser realizada através de um arquivo *.deb* que servirá como instalador para o mesmo.

Capítulo 5

Arquitetura

5.1 Descrição da Arquitetura

O sistema proposto foi idealizado para funcionar localmente no dispositivo N800 da Nokia, funcionando no modo monousuário. O sistema pode ser dividido, em três camadas lógicas que serão conectadas através do padrão MVC [Sauvé 2006], conforme pode ser visto na Figura 5.1:

- **Apresentação:** camada que é responsável por lidar com todo aspecto de interface para interação com o usuário.
- **Aplicação:** camada que é responsável por conter toda a lógica do negócio da aplicação, mantendo, assim total desacoplamento com as demais camadas.
- **Persistência:** camada que é responsável por lidar com toda a persistência de dados de modo que eles possam ser usados posteriormente de forma íntegra.

Vale lembrar que toda a comunicação entre essas camadas, bem como entre os diversos módulos internos as mesmas, é realizada através de um conjunto de interfaces bem definidas, buscando assim, garantir flexibilidade e baixo acoplamento, bem como um eventual reuso e/ou substituição da biblioteca financeira em questão. A separação entre os módulos se deu da forma mais simplificada possível para que, independente da sobrecarga existente, esta não influenciasse fortemente o tempo de resposta desejado para as interações com o usuário da aplicação, este foi um dos requisitos não funcionais especificados na fase de análise. Outro

ponto importante a se destacar é o uso do paradigma OO, bem como do uso de scripts, dada a escolha da linguagem Python para o desenvolvimento da Camada de Aplicação.

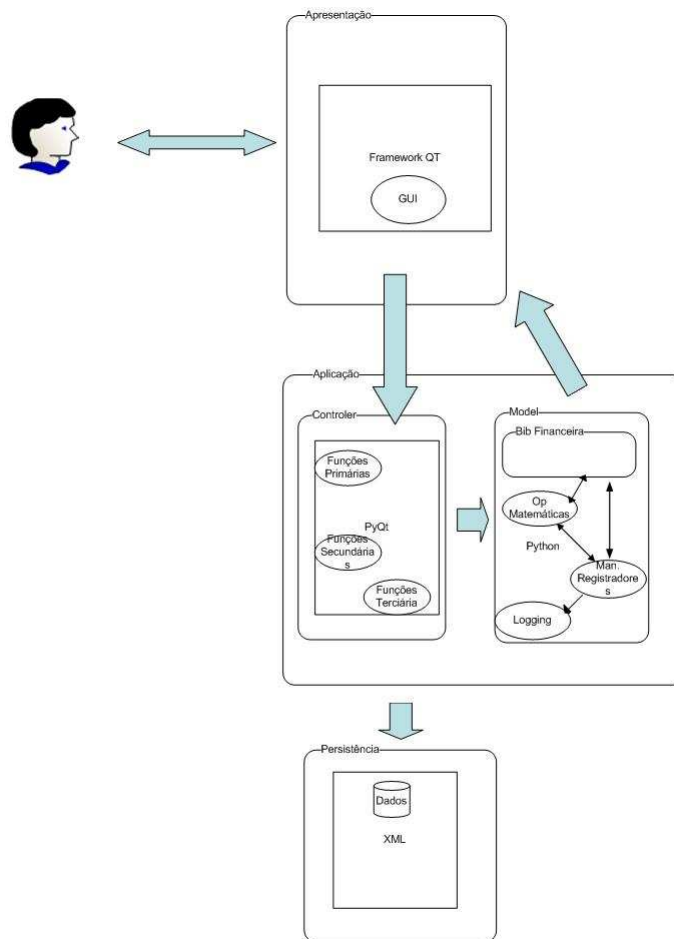


Figura 5.1: Projeto arquitetural da aplicação.

5.1.1 Camada de Apresentação

O sistema é acessado localmente por um único usuário, via uma interface similar à da calculadora financeira que está sendo usada como base, tendo a opção de finalização da aplicação através de um botão. É importante destacar ainda que toda entrada realizada pelo usuário se dá através da interface *touch-screen* disponibilizada pelo dispositivo selecionado. Para o desenvolvimento dessa camada fez-se uso do *Framework QT*, com auxílio da ferramenta de design gráfico *QT Designer*. É de suma importância aqui o desenvolvimento de *packages* no que diz respeito aos componentes visuais da calculadora, por exemplo, os botões da mesma.

5.1.2 Camada de Aplicação

Como dito anteriormente, essa camada é responsável por toda a lógica de negócio do sistema e foi desenvolvida em Python devido a suas vantagens na manipulação de números. Dentre os principais componentes desta camada pode-se dar destaque à biblioteca financeira (também desenvolvida pela equipe com o foco de ser um módulo importável que contenha essencialmente funcionalidades financeiras), ao módulo de operações matemáticas, ao módulo de gerenciamento dos registradores existentes (responsável por fornecer os dados a serem persistidos) e ao módulo de comunicação com a interface da calculadora. Dentro dessa camada pode-se destacar que as interações ocorrem em duas etapas:

1. Numa primeira etapa tem-se a comunicação dessa camada com a camada de apresentação do sistema através dos controladores, fazendo estes uso dos *bindings* em PyQt e sendo subdivididos de acordo com as funcionalidades providas por cada tecla.
2. Numa segunda etapa observa-se:
 - A comunicação dos módulos controladores com o restante da lógica da aplicação, fazendo uso das funcionalidades da biblioteca financeira, do módulo matemático e manipulando os seus registradores.
 - A interação entre os registradores e a biblioteca, onde os primeiros podem fornecer entradas para a última, a interação entre os registradores e o módulo matemático e a interação entre o módulo matemático e a biblioteca.

5.1.3 Camada de Persistência

A persistência dos dados relativos aos registradores é realizada através do uso de arquivos. Os dados que são armazenados nos arquivos são recarregados sempre que houver a inicialização da aplicação, demandando, assim, pouca interação com o disco. Outro aspecto relevante a ser considerado aqui é a segurança do sistema. Devido à natureza do sistema focou-se nos aspectos de integridade dos dados manipulados, fazendo com que os arquivos utilizados fossem armazenados de forma oculta ao usuário, evitando, assim, alterações dos mesmos por parte deste.

5.1.4 Packaging

Foi tomada a decisão de distribuir a aplicação através de um arquivo de instalação de extensão *.deb*, simplificando, assim, a maneira de instalação do software. Tal arquivo, futuramente, poderá ser obtido por download de certo servidor web que será escolhido pela equipe em conjunto com o cliente.

5.1.5 Pontos Adicionais

Com relação à integração com outras aplicações, é do interesse da equipe que a biblioteca financeira desenvolvida possa vir a ser usada por outros desenvolvedores como um dos vários módulos que auxiliem sua aplicação específica. Por fim, todo o desenvolvimento foi guiado pelo padrão de operações que é usado pela HP12-C que é a Notação Polonesa Reversa (ou Notação Polonesa Inversa segundo alguns autores [Wikipedia a]).

Capítulo 6

Validação

Uma das etapas mais importantes dos processos de desenvolvimento de software é aquela referente às atividades de Validação. A maioria dos processos modernos, inclusive XP1, indica que as atividades de teste devem ser executadas paralelamente a codificação do sistema, ou seja, existindo código já deve existir testes para o mesmo. Tal atitude garantirá que os testes do sistema possam evoluir conjuntamente com o crescimento do volume de código.

Pelo contexto do projeto estar baseado principalmente na utilização de fórmulas matemáticas e financeiras, buscou-se fundamentar bem os resultados através de uma bateria, grande em volume e em qualidade, de testes (testes de unidade e de aceitação). Para gerenciamento e utilização sistemática dessa bateria de testes fez-se uso de bibliotecas para execução automática dos mesmos. As bibliotecas utilizadas foram: PyUnit para testes de unidade e PyEasyAccept para testes de aceitação.

Outra preocupação constante durante a construção dos testes da aplicação foi com o nível de tolerância que os resultados obtidos poderiam divergir dos resultados apresentados pela HP-12C. Este foi um dos requisitos não funcionais planejados e, para cumprí-lo, fez-se uso em todos os testes de unidade de uma função especial para averiguar se os resultados dos testes estavam dentro da tolerância pré-estabelecida. Quanto aos testes de aceitação, também existiu a verificação se os resultados retornados estavam dentro da faixa desejada.

Para validação da aplicação, durante as reuniões quinzenais com o cliente do projeto e com o professor Adail Marcos, que leciona a disciplina de Matemática Financeira no curso de Administração da UFCG e que orientou a equipe em relação as fórmulas a serem empregadas, foi possível validar se questões como interface e novas funcionalidades estavam de

acordo com as expectativas dos possíveis clientes do produto final.

Toda esta preocupação com testes, além de objetivar minimizar a quantidade de falhas do produto, tiveram por propósito atestar a qualidade do software, bem como permitir que a evolução do mesmo possa ser realizada sem maiores problemas.

Quanto a atividade de Verificação, esta conceitualmente está diretamente relacionada ao estabelecimento/identificação de corretude em parte de um determinado algoritmo, processo, protocolo, arquitetura, etc, para isso utilizando métodos formais de especificação e prova [conceito definido em discussão interna com o prof. Hyggo Almeida]. Visto isto, pela atividade de verificação tratar-se de um processo mais formal, e como o tempo para desenvolvimento do projeto foi reduzido, acabamos por não realizar esta atividade, considerando-a não adequada ao contexto do projeto em questão.

Capítulo 7

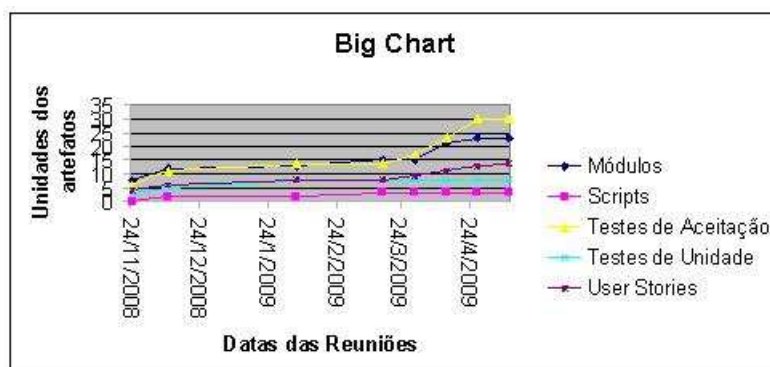
Métricas

Dentre as várias formas que um gerente de projeto tem a sua disposição para acompanhar o andamento de um projeto destaca-se o uso do gráfico *Big Chart*. O mesmo pode ser visto como a análise quantitativa do andamento do projeto. As métricas a serem apresentadas são escolhidas pelo gerente e buscam refletir que pontos deseja-se acompanhar, por exemplo, se número de classes, número de linhas de código, testes de unidade que estão rodando perfeitamente, etc. As métricas escolhidas pela equipe em questão foram:

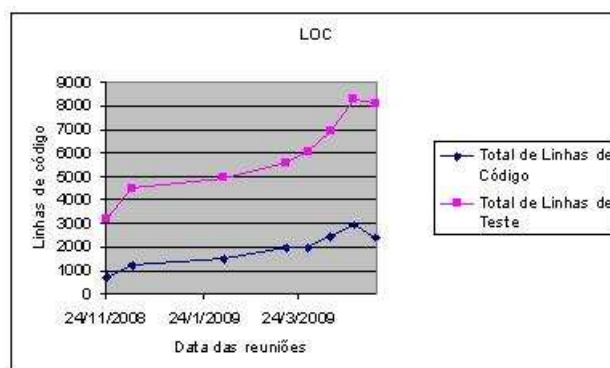
- Número de Módulos python
- Número de scripts
- Número de Scripts de Testes de Aceitação Implementados e Executando Corretamente
- Número de Módulos de Testes de Unidade Implementados e Executando Corretamente
- Número de User Stories Finalizadas
- Total de Linhas de Código
- Total de Linhas de Teste

Vale ressaltar que o *Big Chart* faz parte da documentação especificada em alguns processos vistos na academia como o *YP* [Garcia et al. 2003] e o *XPI* [XP1]. Como o processo de desenvolvimento escolhido durante as disciplinas de Projeto I e Projeto II foi baseado no *XPI*, o *Big Chart* foi utilizado para acompanhamento em alto nível do projeto por parte do cliente, bem como dos integrantes da equipe de desenvolvimento.

Uma vez que as iterações foram definidas com duração de 15 dias e uma *release* contemplou, em média, duas iterações, optou-se em Projeto I por gerar as métricas a cada *release*. Entretanto, como no final da disciplina surgiu um grave problema de convergência no resultado das fórmulas que estavam sendo usadas até então, optou-se por alterar o intervalo de atualização do mesmo, passando agora a atualizá-lo a cada 15 dias. Com isso buscou-se um maior controle sobre o status do projeto.



(a) Métricas Gerais



(b) Linhas de código

Figura 7.1: *Big Chart*.

Conforme pode-se observar nas Figuras 7.1(a) e 7.1(b), o desenvolvimento apresentou-se crescente até os meses de Fevereiro e Março quando o problema acima citado foi encontrado. Nesse instante foi realizado uma pesquisa por novas fórmulas, com teste das mesmas, o que levou a uma estabilidade tanto no total de linhas de código, como no número de módulos, testes de aceitação, testes de unidade e *user stories*. Nesse mesmo intervalo o aumento no total de linhas de teste se deve à especificação de uma maior quantidade de testes de modo a garantir uma maior convergência com os resultados da HP-12C.

Durante o mês de Março, uma vez encontrado um conjunto de fórmulas que propiciou uma convergência adequada, iniciou-se o desenvolvimento das demais *user stories* em questão e por isso explica-se o crescimento das métricas avaliadas. O grande aumento do número de *user stories* se deve ao fato que após a resolução dos problemas com as fórmulas, um conjunto de pequenas *user stories* foram contempladas pelo desenvolvimento. Essas *user stories* tratavam basicamente de funções fáceis de implementar, cujas fórmulas já haviam sido encontradas. Optou-se por deixar a implementação dessas funções por último, porque *XPI* aconselha começar o desenvolvimento pelas funcionalidades mais importantes (nesse caso, mais difíceis de serem implementadas). Pode-se observar que o número de testes de aceitação também aumentou para apoiar as novas *user stories*.

As últimas avaliações realizadas em Maio apresentam valores constantes na Figura 7.1(a) dado que esse foi um período de refatoramento de código e testes, bem como de melhorias em aspectos da interface. É interessante ressaltar que as alterações aqui produzidas levaram a uma certa queda no total de linhas de código e testes, Figura 7.1(b), dado que foi observado certa redundância em trechos de código e teste, bem como o fato de que foram realizadas simplificações na interface, o que leva a um menor número de linhas de código.

Vale observar que o número de scripts não variou muito durante todo o projeto, pois eles foram utilizados basicamente para *deploy* da aplicação. Os scripts de teste estão contabilizados nas métricas de testes de aceitação e de testes de unidade.

Outro ponto que merece explicação é o porquê da quantidade de linhas de teste ser tão superior à quantidade de linhas de código. Mediante o uso do framework QT e da linguagem Python, bem como da busca de uso de boas técnicas de programação e design, o código da aplicação tornou-se bastante enxuto e reduzido. Quanto aos testes, uma vez que buscou-se uma grande similaridade com os resultados da HP-12C, foi elaborada uma alta carga de testes, buscando contemplar a maior variedade de cenários possível no tempo de desenvolvimento proposto.

Capítulo 8

Conclusões e Trabalhos Futuros

Um dos méritos desse projeto foi trabalhar com conhecimento multidisciplinar, envolvendo conhecimentos adquiridos em disciplinas de computação, como Engenharia de Software, Métodos de Software Numérico e Sistemas de Informação I e II, assim como em disciplinas extras, como Princípios de Administração Financeira. Dessa forma, os integrantes da equipe tiveram a oportunidade de exercitar os conhecimentos adquiridos em sala de aula.

A opção por desenvolver uma aplicação para dispositivos móveis também contribuiu para o amadurecimento da equipe no tocante a perceber as particularidades e as dificuldades de se trabalhar no desenvolvimento desse tipo de sistema. Além disso, essa experiência possibilitou o contato com diversas novas tecnologias, aumentando o conhecimento da equipe e inserindo diferenciais nos currículos profissionais.

Uma das dificuldades enfrentadas pelo grupo foi a falta de apoio na área financeira. O cliente cumpriu muito bem o seu papel, porém este tinha maior embasamento na área de dispositivos móveis. Houveram diversas tentativas, por parte do grupo e do cliente, em obter um apoio na área financeira, porém com pouco sucesso.

Como trabalhos futuros, planeja-se a adição de novas fórmulas financeiras, bem como o refatoramento das fórmulas existentes para introduzir métodos melhores para calcular os resultados. Planeja-se ainda trabalhar em melhorias na interface com o usuário, visto que isso não foi possível no decorrer das disciplinas pois a biblioteca gráfica utilizada ainda estava em processo de migração para a plataforma Nokia Maemo.

Enfim, toda a experiência vivida em Projeto I e II possibilitou o crescimento dos alunos como desenvolvedores, analistas e gerentes de projeto, aguçando as habilidades de resolução

de problemas e permitindo o contato com novos conceitos e paradigmas.

Apêndice A

Glossário

A

- Amortização: A palavra amortização pode ser entendida como a diminuição da dívida, de uma única vez ou aos poucos, em mais vezes.

C

- Capitalização: Periodicidade do vencimento do juro ou número de vezes em que o juro é processado (calculado) num ano: anual, semestral, trimestral, mensal, etc.
- Capitalização Linear: A capitalização linear – ou método de formação dos juros simples – considera a incidência periódica de uma dada taxa de juros sobre uma base de cálculo fixa.
- Capitalização exponencial: A capitalização exponencial ou – método de formação dos juros compostos – considera a incidência periódica de uma dada taxa de juros sobre uma base de cálculo variável.

- Capital inicial: Capital existente na data-pólo inicial.
- Capital final: Capital existente na data-pólo final.

D

- Data-pólo inicial: Data de início do investimento.
- Data-pólo final: Data de resgate do investimento.

- Datas-focais intermediárias: Datas localizadas entre a data-pólo inicial e a data-pólo final, onde existem movimentações financeiras.
- Diagrama de tempo: Diagrama que apresenta o fluxo de caixa juntamente com os instantes de tempo das movimentações. Por convenção, recebimentos, entradas ou fluxos positivos são representados por setas apontando para cima ao passo que saídas ou pagamentos ou fluxos negativos são representados por setas apontando para baixo.

F

- Fluxo de Caixa: Montante de caixa recebido e gasto por uma entidade durante um período de tempo definido, algumas vezes ligado a um projeto específico.

J

- Juros de uma prestação em um sistema de amortização: Quantia paga além do valor amortizado em cada parcela. É calculada em função da taxa de juros e do saldo devedor do instante anterior.
- Juros Simples: No sistema de juros simples, os juros são calculados sobre o principal da dívida (valor inicial emprestado ou aplicado).
- Juros Compostos: No sistema de juros compostos, os juros de cada período somam-se à dívida, incidindo juros sobre ele no período seguinte.

P

- Pagamento Antecipado: Pagamento de uma prestação que é realizado no início de um dado período.
- Pagamento Postecipado: Pagamento de uma prestação que é realizado no término de um dado período.
- Prestação: Instrumento de recuperação ou de devolução de um capital acrescido dos juros contratados no período de negociação da transação.

S

- Saldo Devedor: Montante que ainda necessita ser abatido em um dado instante de tempo considerando um certo sistema de amortização.
- Série Simples: Quando um fluxo de caixa apresenta apenas datas-pólo, ou seja, temos apenas dois fluxos.
- Série Complexa: Quando um fluxo apresenta ao menos uma data-focal intermediária além das datas-pólo, ou seja, tem-se no mínimo três fluxos.
- Série Uniforme de Prestações Uniformes: Conjunto de pagamentos ou recebimentos nominais iguais, dispostos em períodos constantes ao longo de uma série complexa de fluxo de caixa. Pode ser de três tipos: postecipada, antecipada ou diferida.
- Série Uniforme de Prestações Uniformes Antecipada: São pagamentos ou recebimentos distribuídos por todos os períodos de uma série complexa de fluxo de caixa, inclusive na data-pólo inicial ou data-focal zero.
- Série Uniforme de Prestações Uniformes Diferida: São pagamentos ou recebimentos distribuídos em uma série complexa de fluxo de caixa por períodos situados após um determinado prazo de carência.
- Série Uniforme de Prestações Uniformes Postecipada: São pagamentos ou recebimentos distribuídos por todos os períodos de uma série complexa de fluxo de caixa, menos na data-pólo inicial ou data-focal zero.
- Sistema de Amortização: É a forma pela qual são calculadas as prestações que você vai pagar no decorrer do financiamento. Nos empréstimos de longo prazo, esses pagamentos são, normalmente, efetuados por parcelas (prestações), compostas de cotas de amortização e de juros.
- Sistema de Amortização Constante: É um método onde as amortizações do capital apresentam comportamento uniforme ou constante durante o período de vigência de uma operação financeira. Os juros apresentam valores heterogêneos e decrescentes ao longo do tempo, o que resulta em prestações heterogêneas e decrescentes ao longo do tempo.

- Sistema de Amortização Mista: Criado em 1979 pelo BNH. Representa um plano de amortização misto, a partir da combinação entre os sistemas Francês e Constante, onde juros, amortização e prestação, derivam de médias aritméticas envolvendo os valores calculados no PRICE e no SAC. Os juros apresentam valores heterogêneos e decrescentes ao longo do tempo. Com os juros heterogêneos e amortizações homogêneas temos uma configuração de prestações heterogêneas e decrescentes.
- Sistema Francês: Criado no século XVIII pelo matemático, filósofo e teólogo inglês Richard Price (por isso o nome Sistema Price), é um método de amortização com prestações iguais e vencidas, onde juros e amortizações portam-se de maneiras decrescente e crescente.

T

- Taxa de Juros: É o percentual que remunera o capital. Uma taxa de juros incide sobre um capital disposto em um fluxo de caixa sempre quando da ocorrência de uma variação temporal, conhecida como frequência intervalar necessária à formação dos juros.

V

- Valor Atual ou Valor Presente: Representa um fluxo qualquer situado na data-pólo inicial.
- Valor Nominal Representa um fluxo qualquer situado entre a data-focal intermediária 1 e a data-pólo final.
- Valor Futuro Antecipado: Representa o somatório de todas as prestações capitalizadas à data-pólo final, com uma mesma taxa de juros; o PMT contido na data-pólo final não sofre nenhuma alteração
- Valor Futuro Postecipado: Representa o somatório de todas as prestações capitalizadas à data-pólo final, com uma mesma taxa de juros.
- Valor Presente Antecipado: Representa o somatório de todas as prestações descontadas à data-pólo inicial, com uma mesma taxa de juros; o PMT data-pólo inicial não sofre nenhuma alteração.

- Valor Presente Postecipado: Representa o somatório de todas as prestações descontadas à data-pólo inicial, com uma mesma taxa de juros.

Apêndice B

Fórmulas

Aqui serão apresentadas as fórmulas usadas, bem como as fontes a partir das quais as mesmas foram obtidas:

1. pv - BEG [Lutus b]:

$$pv = (i + 1)^{-n} * (-fv * i - (i + 1) * ((i + 1)^n - 1) * pmt) / i$$

2. pv - END [Lutus b]:

$$pv = (i + 1)^{-n} * (-pmt * (i + 1)^n - fv * i + pmt) / i$$

3. pv com $i = 0$ [dos Santos 2001]:

$$pv = fv + n * pmt$$

4. fv - BEG [Lutus b]:

$$fv = ((i + 1) * pmt - (i + 1)^n * (i * pmt + pmt + i * pv)) / i$$

5. fv - END [Lutus b]:

$$fv = (pmt - (i + 1)^n * (pmt + i * pv)) / i$$

6. fv com i = 0 [Lutus a]:

$$fv = -(pv + n * pmt)$$

7. n - BEG [Lutus b]:

$$n = \log((-fv * i + pmt * i + pmt) / (i * pmt + pmt + i * pv)) / \log(i + 1)$$

8. n - END [Lutus b]:

$$n = \log((pmt - fv * i) / (pmt + i * pv)) / \log(i + 1)$$

9. n com i = 0 [Lutus a]:

- Se pólos com sinal igual:

$$n = |(pv - fv)| / |pmt|$$

- c.c:

$$n = (|pv| - |fv|) / |pmt|^1$$

10. pmt - BEG [Lutus b]:

$$pmt = -i * (pv * (i + 1)^n + fv) / ((i + 1) * ((i + 1)^n - 1))$$

11. pmt - END [Lutus b]:

$$pmt = -i * (pv * (i + 1)^n + fv) / ((i + 1)^n - 1)$$

12. pmt com $i = 0$ [Lutus a]:

- Se pólos com sinal igual:

$$pmt = |(pv - fv)| / |n|$$

- c.c:

$$pmt = (|pv| - |fv|) / |n|^1$$

13. i: Usa-se a função do fv com estimativas de i ² [Lutus a]

14. npv [HP 2009a]:

$$NPV = CF_0 + CF_1 / (1 + i)^1 + CF_2 / (1 + i)^2 + \dots + CF_n / (1 + i)^n$$

15. irr [Rangel 2009]: Utilizado o método da secante para achar um valor de taxa na fórmula do NPV em que $NPV = 0$.

16. SAF: pmt [Silva 2009]

$$pmt = pv * (1 + i)^n * i / ((1 + i)^n - 1)$$

17. SAF: amort [Silva 2009]

$$A_t = (pmt - (pv * i)) * (i + 1)^{t-1}$$

18. SAC: juros [Silva 2009]

$$J_t = pv * i - (A_t * i * t - 1)$$

19. SAC: pmt [Silva 2009]

$$pmt_t = A_t + J_t$$

20. SAC: amort [Silva 2009]

$$A_t = pv/n$$

21. Conversão do n [HP 2009d]:

$$n_{mensal} = n_{anual} * 12$$

22. Conversão do i (juros simples) [Gondim 2009]:

$$i_{mensal} = i_{anual}/12$$

23. Conversão do i (juros compostos) [Gondim 2009]:

$$i_{mensal} = (1 + i_{anual})^{1/12} - 1$$

24. Percentagem de um dado valor [HP 2009a]:

$$\% = Base(y) * Rate(x) / 100$$

25. Variação Percentual [HP 2009a]:

$$\Delta\% = 100 * (NewAmount(x) - Base(y)) / Base(y)$$

26. Percentagem do Total [HP 2009a]:

$$\%T = 100 * (Amount(x) / Total(y))$$

Observações:

¹ : Faz-se ainda um novo cálculo do pv com o valor resultante do n. Se o valor retornado for diferente, inverte-se o sinal do n.

² : O algoritmo base inicia com uma taxa de juros de 100% e iterativamente, no máximo duas iterações mais externas trocando o sinal da taxa ou até achar a solução busca-se um novo valor de i. Internamente tenta-se acrescentar uma estimativa atual de um valor gd, alterado de 0.5 ou -0.5 de acordo com certas condições, e verifica-se a proximidade do resultado dessa estimativa na função do fv em relação ao valor real do fv. Realizando-se três tentativas consecutivas de cálculo de fv que fiquem com um erro inferior a $1 * 10^{-8}$ para-se o algoritmo, ou então tenta-se um número máximo de 400 iterações internas em estimativas do i.

Apêndice C

Novas Funcionalidades

Dentre as várias características de um software bem projetado, pode-se ressaltar que uma das desejadas é permitir facilmente a inclusão de novas funcionalidades. Para isso elaborase um bom design do mesmo, bem como uma boa documentação. Com esse intuito aqui será apresentado um pequeno guia que serve de auxílio para inclusão de novas funcionalidades, bem como para um melhor entendimento do sistema.

C.1 Adicionando uma função

C.1.1 Considerações Iniciais

Inicialmente serão destacados alguns esclarecimentos básicos a respeito da organização do código fonte:

1. Em relação a interface gráfica, o módulo principal que organiza os dados de interface gráfica da calculadora, incluindo a disposição de botões, é o CalcGui. Além desse, existem módulos que lidam com a tabela de amortização (calcTableDialog), com a tabela de *recall* (recallViewDialog) e com a tabela de *store* (storeViewDialog).
2. Existe um componente que representa um botão (calcButton). Para cada botão pode-se adicionar até 3 textos de funções: uma na parte superior, um na parte inferior e um na parte central.

3. O *controller* (calcController) captura todos os eventos de teclas pressionadas. A identificação das teclas se dá considerando o texto da parte central do botão (que representa a função principal de cada tecla). Em seguida o *controller* chama uma ação correspondente no *model* (calcModel).
4. O calcModel é a lógica de negócio do sistema. Esse módulo guarda o estado da calculadora (opções de arredondamento, se as teclas especiais f ou g estão pressionadas, etc) e conhece a pilha (calcStack), os demais registradores (calcRegs) e as funções da biblioteca financeira (pyFinancialLibrary).
5. O calcModel oferece publicamente funções que casam com as funções tidas como principais de cada botão (a função cujo texto fica no meio do botão). Dado que o calcModel conhece o estado da calculadora, em cada uma dessas chamadas verifica-se que função da biblioteca usar analisando, por exemplo, se o usuário selecionou os botões F ou G.
6. Na ligação da lógica de negócio, da interface gráfica e do controlador foi utilizado o modelo MVC [Sauvé 2006]. Os módulos da interface gráfica atuam como *Observers* da lógica de negócio e assim são ativados pela mesma quando necessários. O model tem vários *listeners* para eventos como: modificação do valor da tela, modificação da pilha, etc. A ativação de cada um desses *listeners* ocorre ao final de cada função do model, bem como a análise de outras ações necessárias como, por exemplo desativar a seleção de F ou G feita pelo usuário.
7. Os dados fornecidos de volta aos *listeners* estão sempre no tipo *Decimal*, que é o tipo trabalhado pela calculadora de modo a alcançar alta precisão. Cada módulo da GUI formata esses dados como queira, fazendo uso das funções de formatação presentes no módulo calcFormatUtil.

Um complemento do que foi exposto aqui pode ser analisado avaliando-se os diagramas de classes presentes no Apêndice E.

C.1.2 Exemplo de Adição de Função

De modo a esclarecer melhor os pontos acima levantados, será simulada a implementação de uma nova funcionalidade na calculadora. Como exemplo, tem-se a adição de uma nova forma de divisão em que ao invés de se dividir o registrador Y pelo X, irá se dividir X por Y. Através desse exemplo, pretende-se abordar o passo-a-passo para se implementar uma nova funcionalidade no sistema em questão, considerando o caso simples de adição de uma função como principal em um dado botão.

Primeiramente é preciso escolher um botão do teclado da GUI que esteja livre para receber a nova função, ou ainda substituir uma função já alocada a algum botão. No método

```
__createKeyboardLayout
```

Em seguida, deve-se criar a ação a ser disparado no clique do botão. Para isso adiciona-se uma nova verificação do label central do botão pressionado no método **keyboardButtonClicked(self, text)** do *controller*. Um exemplo pode ser observado no trecho de código C.1:

Listagem C.1: Teste no controller

```
elif text == "INV /":  
    self.__model.invDiv()
```

No *model* deve-se criar a função chamada no *controller*, conforme visto acima. Na nova função do *model* implementa-se a funcionalidade considerando o estado dos botões F e G, os valores dos registradores, o estado da calculadora e as funções disponíveis na biblioteca de funções (pyFinancialLibrary).

Com o resultado da divisão em mãos, altera-se o valor de algum(ns) dos registradores conforme necessário. Conforme dito acima, ao final de cada função deve-se, se necessário, desativar as teclas F e G, alterar a posição de entrada de decimais, e conforme o que for modificado (tela, pilha, registradores financeiros, etc) avisar aos devidos *listeners* os novos valores que lhes são de interesse.

Um ponto importante é também alterar o modo de entrada de dados da calculadora de acordo com a funcionalidade desejada. Existem basicamente três modos, explicitados na classe *Mode*: *Mode.SaveMode* (que na próxima entrada de dados devolve o valor 0.0 a tela

e então inicia o processamento do que for entrado), *Mode.OperationMode* (devolve o valor zero a tela, mas antes dá um enter na pilha) e *Mode.EntryMode* (indicando que novos dígitos estão sendo entrados ao valor anterior da tela).

Um exemplo de implementação da nova função de divisão no calcModel pode ser observado no trecho de código C.2.

Listagem C.2: Definição de Função no Model

```
def invDiv(self):
    if self.__isGPushed or self.__isFPushed:
        self.__fireException("Function not yet implemented.")
        self.__deactiveFandG()
5    else:
        # "inv /"
        try:
            result = self.__calcStack.getXReg() / self.__calcStack.
                getYReg()
            self.__calcStack.rollCounterClockWise()
10        self.__calcStack[0] = result
            self.__calcStack[3] = self.__calcStack[2]

            self.mode = Mode.OperationMode
            self.base = None
15        self.dotActivated = False

            self.__fireStackRegisters(self.getAllStackRegisters())
            self.__fireScreenChangedEvent()

20        except Exception, e:
            self.__fireException(e.message)
```

Apêndice D

Tabela de Erros

Como sabe-se sistemas apresentam restrições quanto as entradas fornecidas de modo a realizar determinadas atividades. Na HP-12C entradas inválidas produzem erros que são apresentados ao usuário através de um código no formato **ERROR d+**, onde d+ indica um ou mais dígitos. Esse formato dificulta o entendimento por parte do usuário do motivo que levou ao erro, ou seja, o usuário necessita ter em mãos o manual da calculadora que explicita o significado do código visualizado.

Nesse sentido, buscou-se fazer uso das facilidades que o dispositivo computacional em foco oferece de modo a melhorar a usabilidade do sistema. Para isso foi realizado um mapeamento dos códigos de erros encontrados nos cálculos financeiros na HP-12C para mensagens de texto que explicitem melhor o problema encontrado.

Como a aplicação desenvolvida é composta de uma biblioteca financeira que é utilizada pela aplicação calculadora financeira, optou-se por realizar dois mapeamentos: um das mensagens que são geradas pelos cálculos financeiros da biblioteca e outro das mensagens que são geradas por estados inconsistentes na calculadora. Tais mapeamentos estão presentes nas tabelas D.1 e D.2.

Mensagem	Descrição	Correspondente na HP12c
"Invalid scenario: Impossible operations."	Valores declarados tornam impossível a execução da operação, seja por tipos incompatíveis, divisão por zero, etc.	Error 0
"Invalid scenario: Should select BEG or END."	Seletores BEG ou END não selecionados.	-
"Invalid scenario: i value equal or less than -100%."	Valor de i menor que -100.	Error 5
"Invalid scenario: Infinite n."	Valor de n grande o suficiente pra ser considerado infinito.	-
"Invalid scenario: Error(s) in the values of operators capitalization (n, i, PV, FV or PMT)."	Sinaliza erro na operação de capitalização composta, como erros no uso de sinais - e + nos valores monetários (n, i, PV, FV ou PMT) ou algum destes com valor 0 quando isso não é permitido.	Error 5
"Some values for calculating amortization have not been provided."	Alguns dos valores necessários para o cálculo da amortização não foram fornecidos.	Error 7
"Error in some operands to calculate the percent difference."	Valores apresentados para o cálculo da variação de porcentagem são inválidos.	Error 0
"Error in some operands to calculate the percent total."	Valores apresentados para o cálculo da porcentagem total são inválidos.	Error 0

Figura D.1: *Mapeamento de Erros da Biblioteca*

Mensagem	Descrição	Correspondente na HP12c
"All cash flow registers already fulfilled!"	Impossível a inclusão de novos valores, os registradores estão todos preenchidos.	-
"There are only " + Num + " cash flows occurrences"	Existem poucos fluxos de caixa definidos para o cálculo.	-
"Nj must be an integer value between 1 and 99"	Valor estabelecido para Nj fora dos limites.	Error 6
"Invalid cash flow index!"	Número do fluxo solicitado em desacordo com a quantidade de registradores.	-
"Invalid amortization plan!"	Índice escolhido não corresponde a nenhum dos planos de amortização possíveis.	-
"Invalid number pushed: "+ NUM	Número digitado considerado inválido	-

Figura D.2: *Mapeamento de Erros da Calculadora*

Apêndice E

Diagrama de Classes

Aqui temos a visualização dos principais componentes da calculadora financeira apresentados em um diagrama de classes. Os principais módulos e pacotes estão apresentados nas Figuras E.1 e E.2.

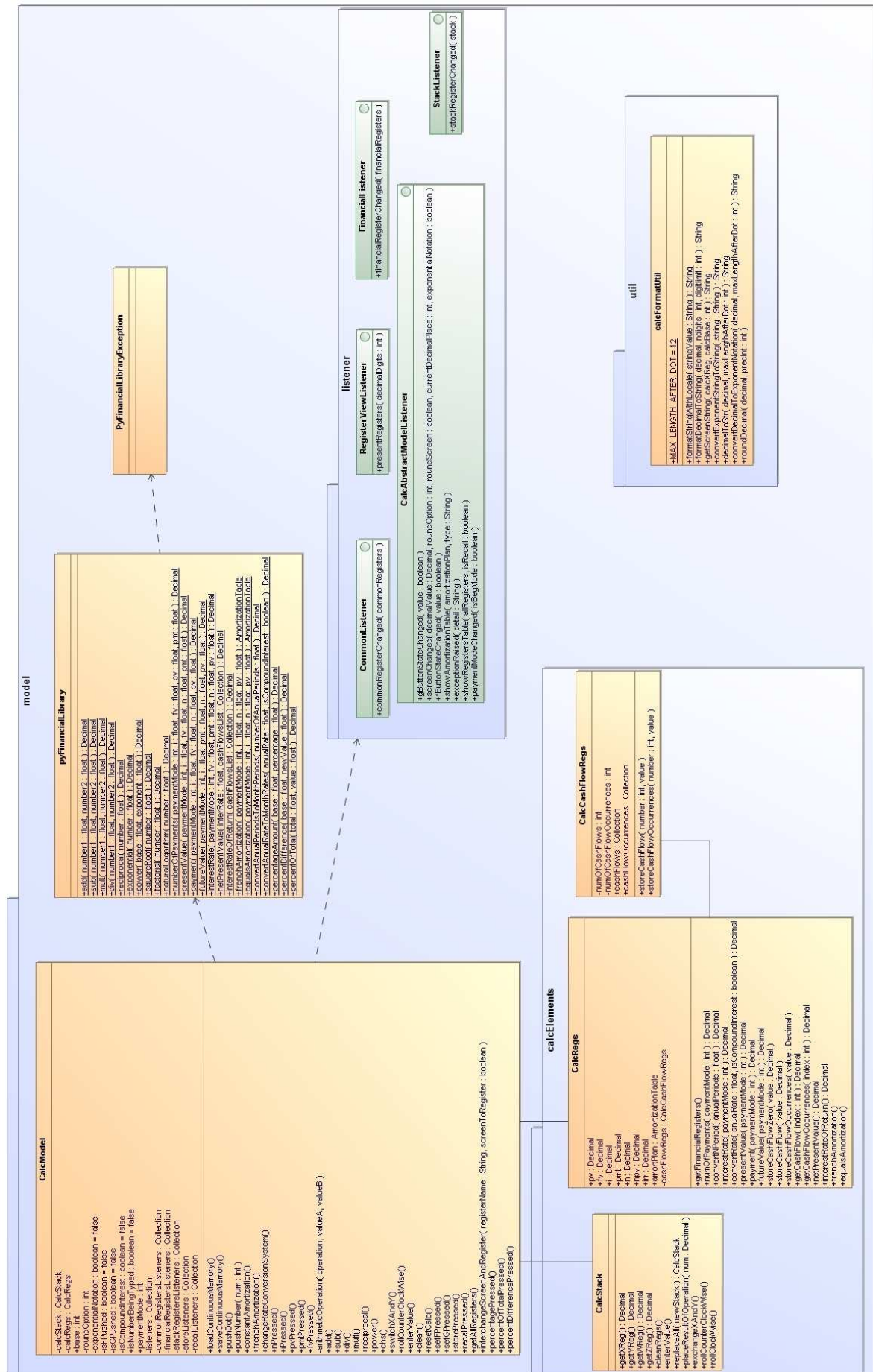


Figura E.2: Pacote model.

Apêndice F

Telas da Aplicação

F.0.3 Tela Principal



Figura F.1: Tela principal da calculadora.

Tela principal da aplicação que permite ao usuário ter acesso a todas as funcionalidades implementadas pela calculadora. É através do mostrador da calculadora que o usuário irá receber o resultado da maioria das funções.

F.0.4 Tela de Exceção

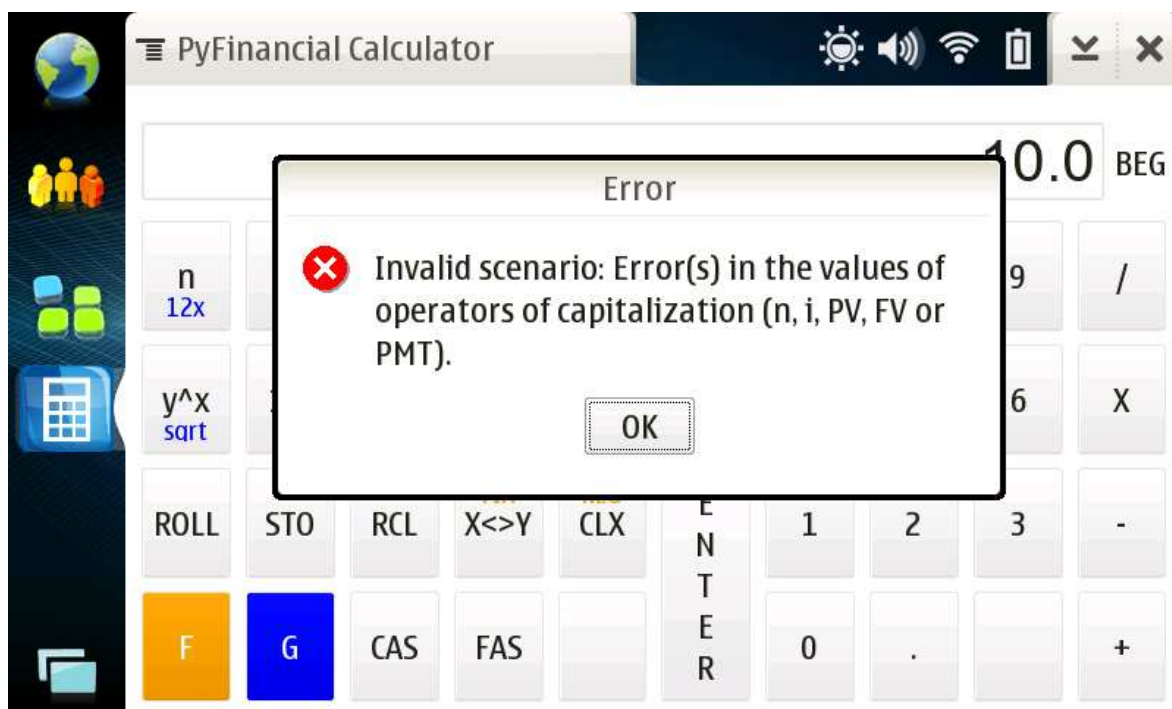


Figura F.2: Tela de exceção.

Tela principal após uma exceção ser gerada. Um *box* é aberto para mostrar ao usuário a mensagem de erro.

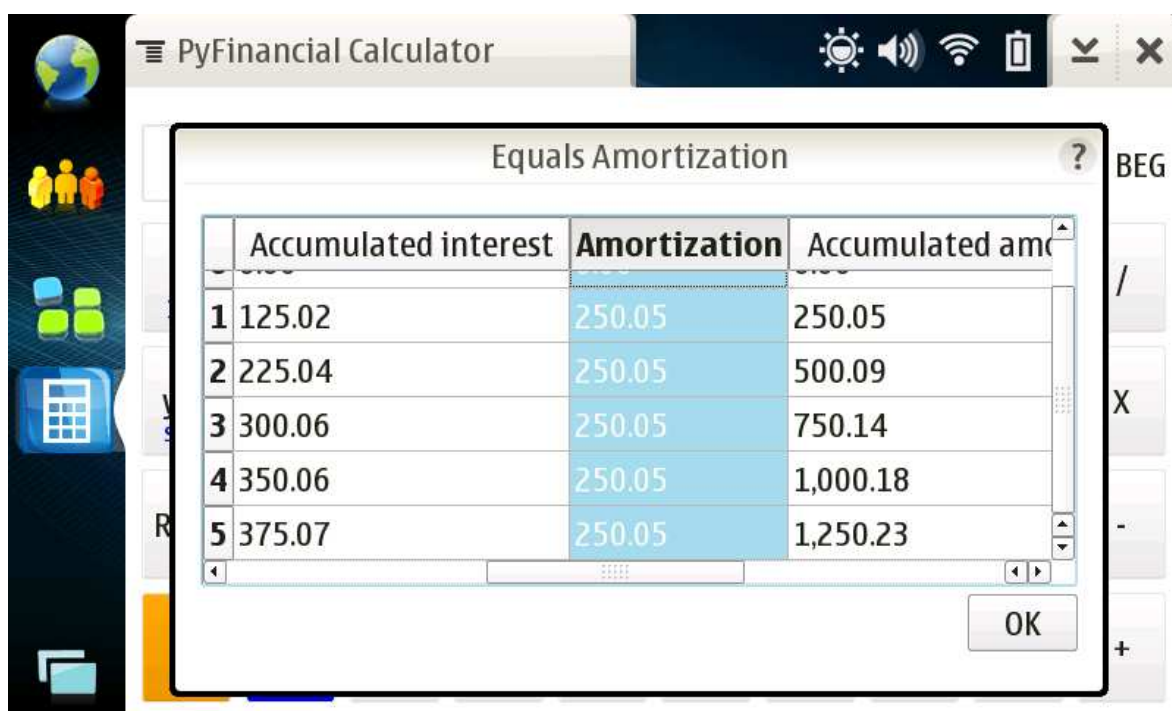
F.0.5 Telas de Amortização

As telas de amortização apresentam no formato de tabela o resultado de um cálculo de amortização constante ou francesa.

F.0.6 Telas de Recall

A tela de *Recall* serve para o usuário selecionar um registrador da calculadora e o valor dele será mandado para o mostrador quando o mesmo clicar em *OK*. Dessa forma, a pessoa poderá utilizar o valor de um registrador em novos cálculos.

Existe também uma tela de *Store*, com o mesmo design da tela de *Recall*, que permite que o usuário faça o inverso, ou seja, armazene o valor do mostrador em um registrador.



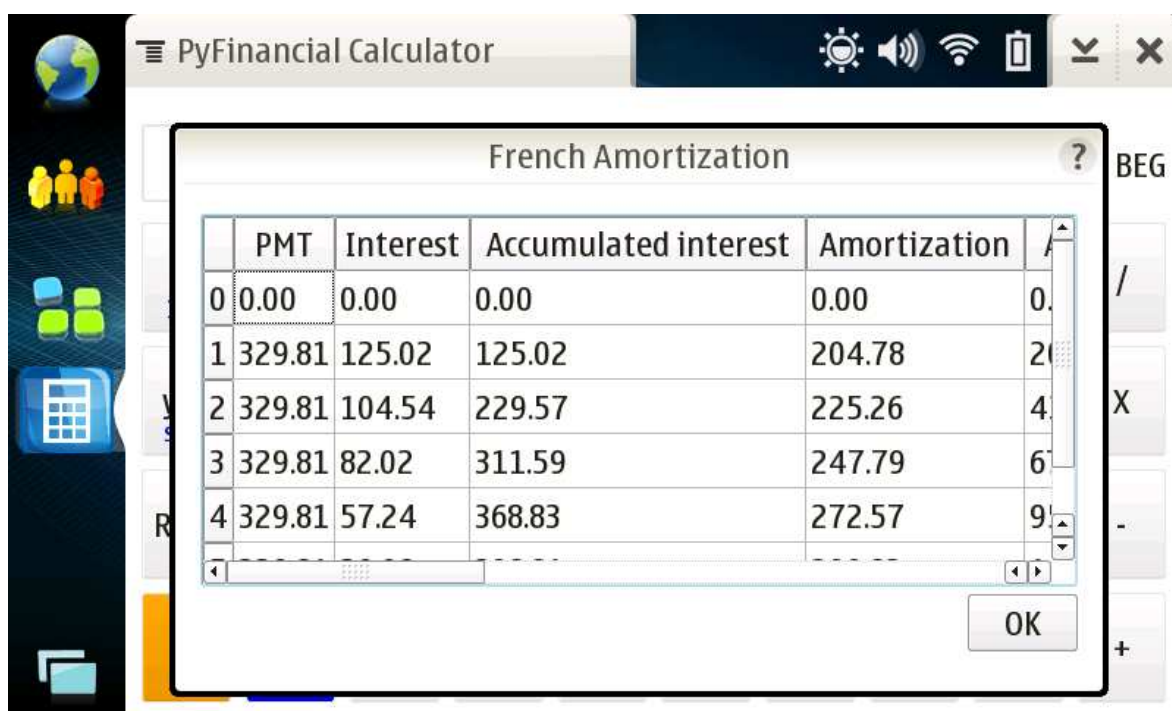
PyFinancial Calculator

Equals Amortization

	Accumulated interest	Amortization	Accumulated amount
1	125.02	250.05	250.05
2	225.04	250.05	500.09
3	300.06	250.05	750.14
4	350.06	250.05	1,000.18
5	375.07	250.05	1,250.23

OK

Figura F.3: Tela de amortização constante.



PyFinancial Calculator

French Amortization

	PMT	Interest	Accumulated interest	Amortization	
0	0.00	0.00	0.00	0.00	0.00
1	329.81	125.02	125.02	204.78	204.78
2	329.81	104.54	229.57	225.26	434.83
3	329.81	82.02	311.59	247.79	682.62
4	329.81	57.24	368.83	272.57	955.19

OK

Figura F.4: Tela de amortização francesa.

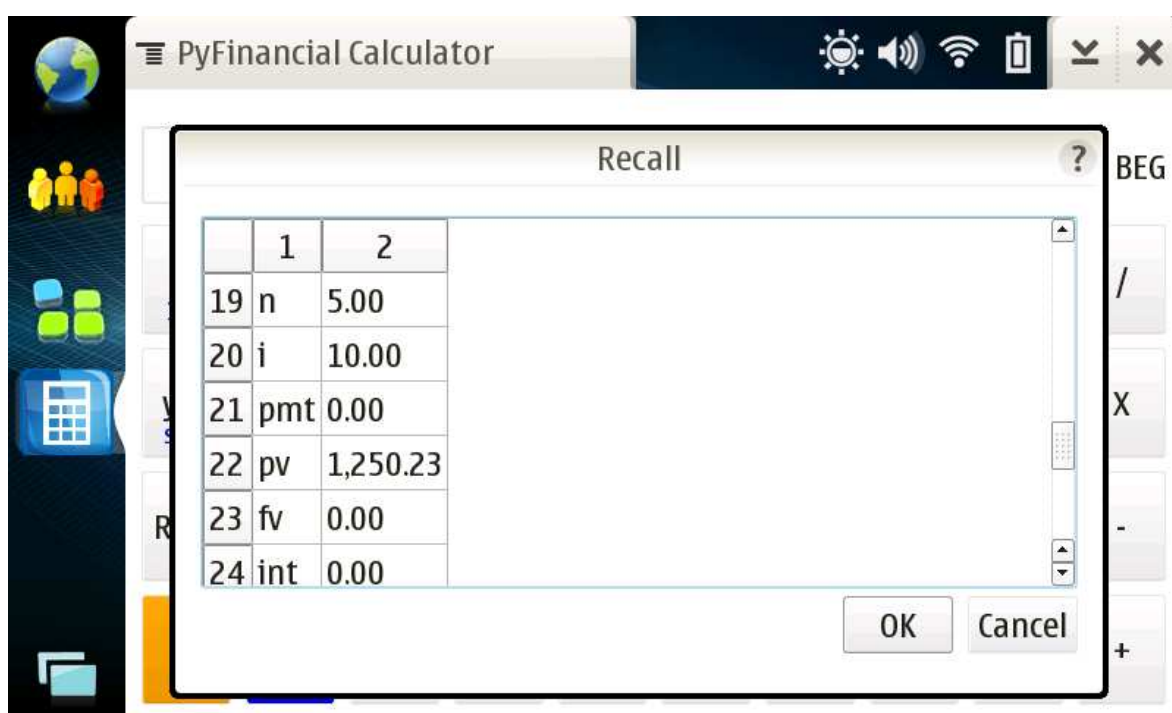


Figura F.5: Tela de recall de registradores.

Bibliografia

[Aptana] Aptana. Disponível em:<<http://pydev.sourceforge.net>>. Acesso em: 01 abr. 2009.

[Azevedo] Azevedo, R. K. A RELEVÂNCIA DA MATEMÁTICA FINANCEIRA NO ENSINO MÉDIO.

[Beck 2003] Beck, K. (2003). *Test-Driven Development by Example*. Addison Wesley.

[dos Santos 2001] dos Santos, J. C. (2001). *Matemática Financeira*. Villipress.

[Educação 2009] Educação, V. (2009). Curso planejamento financeiro pessoal. Disponível em:<<http://www.valoreducacao.com.br/home/a-empresa/38-gerais/48-planejamento-financeiro-pessoal/>>. Acesso em 24 Março 2009.

[FarsightSoft.com] FarsightSoft.com. Disponível em:<<http://mobilecalculator.org/>>. Acesso em 25 Março 2009.

[Fors] Fors, T. Disponível em:<<http://code.google.com/p/hpcalc-iphone/>>. Acesso em 25 Março 2009.

[Foundation] Foundation, E. Disponível em:<<http://www.eclipse.org/>>. Acesso em: 01 abr. 2009.

[G1 2009] G1 (2009). Saiba o que mudou no celular, o aparelho mais usado no brasil. Disponível em:<<http://g1.globo.com/Noticias/Tecnologia/0,,MUL1053098-6174,00-SAIBA+O+QUE+MUDOU+NO+CELULAR+O+APARELHO+MAIS+USADO+NO+BRASIL.html>>. Acesso em 24 Março 2009.

- [Garcia et al. 2003] Garcia, F., Ferreira, A., Sousa, D., Júnior, F., Rocha, G., Mendes, G., Pontes, R., Rocha, V., Dantas, V., and Aguiar, Y. (2003). *easyprocess: Um processo de desenvolvimento de software para uso no ambiente acadêmico*.
- [Gondim 2009] Gondim, C. L. (2009). Juros compostos e taxas equivalentes. Disponível em:<<http://groups.google.com/group/tecprincipios-de-administracao-financeira-20091/files>>. Acesso em: 02 maio 2009.
- [HP] HP. Disponível em:<<http://www.hp.com/latam/br/produtos/calculadoras/12c.html>>. Acesso em: 30 mar. 2009.
- [HP 2009a] HP (2009a). Hp 12c financial calculator user's guide. Disponível em:<[http://h10025.www1.hp.com/ewfrf/wc/manualCategory?lc=en&dlc=en&cc=us&lang=en&product=315564&#/>. Acesso em 24 Março 2009.](http://h10025.www1.hp.com/ewfrf/wc/manualCategory?lc=en&dlc=en&cc=us&lang=en&product=315564&#/)
- [HP 2009b] HP (2009b). Hp 12c internal rate of return. Disponível em:<<http://h20331.www2.hp.com/Hpsub/cache/299940-0-0-225-121.html>>. Acesso em 24 Março 2009.
- [HP 2009c] HP (2009c). Hp 12c net present value. Disponível em:<<http://h20331.www2.hp.com/Hpsub/cache/299940-0-0-225-121.html>>. Acesso em 24 Março 2009.
- [HP 2009d] HP (2009d). Hp 12c platinum calculadora financeira manual do usuário. Disponível em:<<http://www.hp.com/ctg/Manual/bpia5184.pdf>>. Acesso em 24 Março 2009.
- [ISO] ISO. Disponível em:<<http://www.iso.org>>. Acesso em: 01 abr. 2009.
- [Lutus a] Lutus, P. Disponível em:<<http://vps.arachnoid.com/finance/>>. Acesso em: 25 mar. 2009.
- [Lutus b] Lutus, P. Finance calculator version 4.2. Disponível em:<<http://www.arachnoid.com/lutus/finance.html>>. Acesso em: 25 mar. 2009.

- [Mac 2009] Mac (2009). Disponível em:<<http://mac.softpedia.com/get/iPhone-Applications/Tools-Utilities/HP-12C-iPhone.shtml/>>. Acesso em 24 Março 2009.
- [Maemo.org a] Maemo.org. Disponível em:<http://maemo.org/maemo_release_documentation/maemo4.1.x/Maemo_Diablo_Reference_Manual_for_maemo_4.1.pdf/>. Acesso em 25 Março 2009.
- [Maemo.org b] Maemo.org. Garage. Disponível em:<<https://garage.maemo.org/>>. Acesso em: 01 abr. 2009.
- [Maemo.org c] Maemo.org. Maemo qt4. Disponível em:<<http://qt4.garage.maemo.org/>>. Acesso em 25 Março 2009.
- [Maemo.org d] Maemo.org. Pyqt4 for maemo. Disponível em:<<http://pyqt.garage.maemo.org/>>. Acesso em 25 Março 2009.
- [Meirelles 2009] Meirelles, M. (2009). Sistemas de amortização de empréstimos e financiamentos. Disponível em:<http://www.fadepe.com.br/restrito/conteudo/2_adm_matem_finan_sac_saf_saa.doc>. Acesso em: 01 maio 2009.
- [Nokia a] Nokia. Disponível em:<<http://europe.nokia.com/nseries/>>. Acesso em 25 Março 2009.
- [Nokia b] Nokia. Disponível em:<<http://www.s60.com/life>>. Acesso em 25 Março 2009.
- [Pfützenreuter] Pfützenreuter, E. Web hp-12c emulator. Disponível em:<<http://www.epx.com.br/ctb/hp12c.php>>. Acesso em: 25 mar. 2009.
- [Purcell] Purcell, S. Pyunit - the standard unit testing framework for python. Disponível em:<<http://pyunit.sourceforge.net/>>. Acesso em: 01 abr. 2009.
- [python.org] python.org. Disponível em:<<http://www.python.org/download/releases/2.5/>>. Acesso em 25 Março 2009.

- [Rangel 2009] Rangel, E. (2009). Resolução numérica de equações. Disponível em:<<http://www.dsc.ufcg.edu.br/~rangel/msn/downloads/MSN%20NA04.ppt/>>. Acesso em 11 Maio 2009.
- [Sauvé 2006] Sauvé, J. (2006). Model-view-controller (mvc). Disponível em:<<http://jacques.dsc.ufcg.edu.br/cursos/map/html/arqu/mvc/mvc.htm>>. Acesso em: 11 maio 2009.
- [Sauvé et al.] Sauvé, J., Queiroz, M., and Farias, G. Python easyaccept. Disponível em:<<http://www.pyeasyaccept.org/>>. Acesso em: 01 abr. 2009.
- [Silva 2009] Silva, A. M. L. D. (2009). Módulo i - mat. financ. na hp-12c. Disponível em:<http://www.ufcg-uaac.com/Curso_extensao_gestao_investimentos.htm>. Acesso em 24 Março 2009.
- [Tigris.org] Tigris.org. Subversion. Disponível em:<subversion.tigris.org/>. Acesso em: 01 abr. 2009.
- [Wikipedia a] Wikipedia. Notação polonesa inversa. Disponível em:<http://pt.wikipedia.org/wiki/Nota%C3%A7%C3%A3o_polonesa_inversa>. Acesso em: 01 abr. 2009.
- [Wikipedia b] Wikipedia. Taxa interna de retorno. Disponível em:<http://pt.wikipedia.org/wiki/Taxa_interna_de_retorno>. Acesso em: 02 maio 2009.
- [Wikipedia c] Wikipedia. Valor presente líquido. Disponível em:<http://pt.wikipedia.org/wiki/Valor_presente_l%C3%ADquido>. Acesso em: 02 maio 2009.
- [XP1] XP1. Xp1: Um processo de desenvolvimento. Disponível em:<<http://www.dsc.ufcg.edu.br/~jacques/cursos/2002.2/projii/xp1/xp1.html>>. Acesso em: 01 abr. 2009.