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

Conteúdo

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 manido 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], muito disso se deve a 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 a 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, quando partimos para analisar usuários que usam alguns dos conceitos financeiros mas não são especialistas é comum observar-se uma certa rejeição a mesma devido a 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 conforme dito acima.

Observando um pouco a história recente dos dispositivos móveis percebemos 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 disjuntos 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, etc.

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, enxergamos aqui uma boa oportunidade de alavancar um pouco mais esse nicho. Para tal, teremos como objetivo do nosso projeto o desenvolvimento de dois artefatos: i) uma biblioteca de funções, escritas em Python, que implementem as principais funções fincanceiras existentes na HP-12C. Com tal resultado estaremos proporcionando um auxílio para o desenvolvimento de novos aplicativos na área financeira, auxílio este até então inexistente de acordo com nossas pesquisas; e ii) uma calculadora financeira para os dispositivos da Nokia, mais especificadamente para o N800, a qual chamamos 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, nossa calculadora é um software com propósitos financeiros que irá oferecer ao usuário o cálculo das várias funções financeiras básicas existentes na HP-12C, com análises através de juros simples e compostos, 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.

1.1 Contexto do Projeto

1.1.1 Pessoas e Processos

Conforme relatado acima, o perfil principal de usuários que visamos abranger com nossa aplicação é aquele grupo de usuários que não possuem conhecimentos aprofundados em matemática financeira e que consequentemente necessitam de uma interface mais amigável para a realização dos cálculos.

Entretanto, não podemos descartar, devido a facilidade de interação com a aplicação que desejamos, aqueles usuários que desejam aprender a respeito de matemática financeira. Esses podem encontrar em nossa 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, podemos 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 buscamos 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) [Org].
- Ambiente gráfico QT4 [Maemo a] e PYQT4 [Maemo b]
- Python 2.5 [python.org].

É importante destacar que não foram realizados testes com nenhuma versão posterior as que foram acima relatadas por motivos, dentre os quais, a não estabilidade de algumas dessas versões.

1.1.3 Aplicações Correlatas

Nesta sub-seção iremos relatar um pouco a respeito de algumas aplicações similares a calculadora PyFinancial:

- *Mobile Financial Calculator V1.0* [FarsightSoft.com]: Essa calculadora se assemelha a nossa na proposta da facilidade de uso, fazendo uso 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 por nossa aplicação 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 a acurácia de seus resultados. Comparamos resultados oferecidos pela mesma com resultados da HP-12C original, e percebemos distorções consideráveis em certo conjunto de cálculos, por exemplo nos cálculos do pagamento e número de períodos.

- *Finance Calculator Version 4.2* [Lutus a]: 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. Dentre deste ramo, considerou-se na implementação da *PyFinancial* os principais conceitos financeiros existentes na HP-12C **??**.

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 será feita referência às fórmulas que estão implementadas e que se encontram no apêndice **??**.

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 ser 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 de 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 **??**.

2.1.2 Valor Presente - VP ou PV do inglês

Aplicações têm início com uma quantia a partir de um marco inicial. Essa quantia corresponde a quanto vale o investimento inicialmente, sem análises do futuro. A esse valor, que pode ser um valor gasto ou recebido, damos 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 ou do inglês NPV

Uma vez que o valor presente está definido, bem como as prestações, juros da movimentação e número de perídos durante o qual a aplicação estaŕa vigente, uma análise bastante comum é verificar qual o valor da aplicação atualmente. Ou seja, para cada prestação desconta-se o juros existente, seja este simples ou composto, considerando a quantidade de períodos até que a prestação fosse paga e obtém-se o valor da prestação na data-pólo inicial. Em seguida soma-se os valores presentes de cada prestação ao valor presente e obtem-se o NPV.

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

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 ou FV do inglês

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 dessa aplicação. Ou seja, o valor futuro é o resultado da incidência dos juros e das prestações em cima do valor presente. Esse valor também é conhecido como capital final.

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

2.1.5 Taxa de juros - i e Taxa interna de retorno - irr

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 torno 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 o IRR 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]

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, prestações.

2.1.7 Prestações - pmt

As prestações indicam entradas ou saídades 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, se pensarmos 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 considerada 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 define o valor das prestações que inclui tanto um montante a ser abatido do saldo devedor, bem como os juros a serem pagos. Assim, podemos 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 podemos 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 [Wikipedia c].

Capítulo 3

Processo de Desenvolvimento

Conforme as boas práticas que a Engenharia de Software apregoa, 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 (também) como parâmetro de classificação de qualidade, a verificação de quais processo de software são empregados nas instituições.

Sabido da grande importância que um processo de desenvolvimento exerce sobre a qualidade do produto final, buscamos um processo, dentre os existentes na literatura especializada, que mais se adequasse ao contexto do nosso projeto. Dentre os processos existentes decidimos escolher o XP1 [XP1], os motivos que nos levaram a escolha deste ante os demais processos exitentes 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) por tratar-se de um processo desenvolvido por um conjunto de alunos e professores da nossa universidade, tal fato nos trouxe a segurança que as possíveis dificuldades que aparecessem poderiam ser facilmente dissipadas pois estaríamos 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.

XP1 define um conjunto de papéis que os membros da equipe deverão assumir no decor-

rer do desenvolvimento, bem como quais as 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 ***antes*** 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 freqüentemente 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 • 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 anteriomente (cliente, desenvolvedor e gerente) no contexto do nosso projeto tivemos a seguinte classificação: i) como cliente tivemos o professor Dr. Hyggo O. de Almeida; 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 a alocação de atividades (descritas na tabela 3.1), houve sempre a preocupação em dividi-las igualitariamente entre os membros da equipe. Para tal, reuniões de acompanhamento foram realizadas semanalmente. Nessas reuniões os membros da equipe procu-

ravam 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 nosso projeto tivemos a seguinte configuração:

- A atividade de planejamento consumiu bastante tempo ainda na primeira parte da disciplina de Projeto I, pois esta etapa envolveu uma ampla pesquisa na literatura especializada para verificação de quais requisitos, funcionais e não funcionais, deveriam estar presentes no nosso 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 pricipalmente nas ideias de MVC, porém atendeu as nossas necessidades. 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 que 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 desenvolvidos em sequência. E, seguindo o que indica a metodologia de testes TDD [Beck 2003], os testes de unidade foram sempre desenvolvidos antes que a codificação fosse realizada. É importante destacar que como o projeto foi proposto para um cliente que não era da área finan-

ceira, os testes de aceitação foram escritos pelos desenvolvedores 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 Me Adail Marcos Lima Da Silva [Silva 2009], e comparando-se os resultados com os resultados da HP-12C real. Outro ponto importante é que existiu sempre a preocupação de que pessoas diferentes escrevessem e revisassem os testes, com isso procuramos garantir uma maior credibilidade aos nossos testes.

- A atividade de integração foi uma preocupação constante da nossa equipe desde a fase de planejamento. Para que não ocorressem divergências ou inconsistências de conteúdo, tivemos a preocupação de criar um controle de versões tanto para a biblioteca financeira quanto para a calculadora *Pyfinancial*, para tal usamos as já muito conhecidas ferramentas de controle de versões SVN [SVN] e Garage [garage], respectivamente. Com isso, conseguimos manter um controle sistematizados do nosso código, bem como dos nossos artefatos de planejamento.

- Quanto a realização de *Reviews*, sempre foi alocado 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 tenhamos procurado organizar o design de maneira flexível, percebemos no decorrer do projeto que alguns pontos poderiam sofrer um refatoramento maior. Essa necessidade possivelmente está relacionada com o fato de não termos alocado desde o início um tempo maior para *Code* e *Design Review*.

- Por fim, como já dito anteriormente, a atividade de gerência foi revesada 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 *realises* 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 afim de melhorar a organização e consequentemente a qualidade do código produzido. Dentre os elementos presentes nessa infra-estrutura podemos destacar:

- **Eclipse** [eclipse]. IDE de desenvolvimento.
- **PyDev** [pydev]. *Plug-in* para desenvolvimento em Python.
- **PyUnit** [pyunit]. *Framework* para desenvolvimento de testes de unidade.
- **PyEasyAccept** [pyeasyaccept]. *Framework* para desenvolvimento de testes de aceitação.
- **SVN e Garage SVN** [SVN][garage]. Ferramentas para controle de versões

Capítulo 4

Análise de Requisitos

4.1 Modelo Conceitual

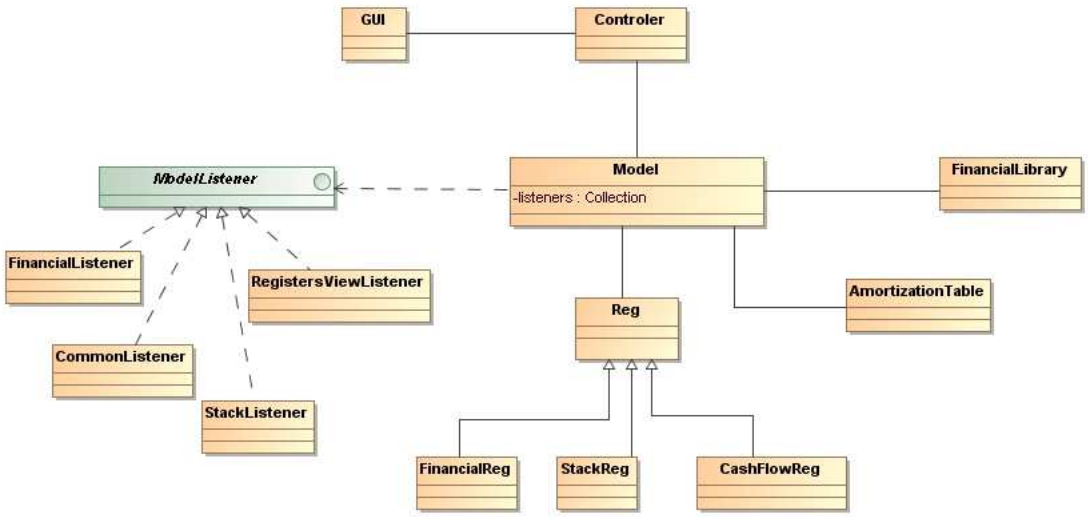

Figura 4.1: *Tabela que descreve quando as atividades de XP1 devem ser realizadas.*

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

- Model
- GUI
- Controler

- AbstractModelListener
- Reg
- FinancialLibrary

4.1.1 Model

Módulo central da aplicação que será responsável por fazer a troca de mensagens entre as demais entidades do sistema as quais está conectada, bem como realizar toda a manipulação de registradores (pilha, 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 estará em contato direto com o usuário da aplicação. Logo, este é o módulo que receberá os dados externos e os transmitirá para entidades mais internas, assim como receberá os resultados das operações realizadas pela calculadora e os transmitirá ao cliente de uma maneira mais amigável. Essa interface também é responsável por apresentar os dados segundo preferências do usuário previamente selecionadas, por exemplo, com um dado número 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. Receberá as requisições feitas pelo cliente da aplicação, através da interface (*GUI*), e identificará as operações correspondentes no *Model* que se responsabilizarão por realizar as atividades correspondentes.

4.1.4 AbstractModelListener

Entidade abstrata da qual outras a generalizarão. Estas serão responsáveis por capturar qualquer mudança importante ocorrida no *Model* que seja de interesse do usuário. Assim essas alterações serão refletidas na interface. Aqui encaixa-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 armazena-se os valores a serem utilizados nas operações, bem como os resultados das operações requisitadas. Esta entidade será 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 consequentemente, 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 as reuniões de planejamento entre os membros da equipe, o cliente e possíveis usuários dos nossos produtos, um conjunto de requisitos funcionais e não funcionais ficaram estabelecidos. São eles:

- **Requisitos Funcionais**
  - O usuário deverá ser capaz de realizar as operações matemáticas/financeiras seguindo a Notação Polonesa Reversa [NPR].

- 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 present (PV), valor da parcela (PMT) e valor futuro (FV). Esses cálculos podem envolver ou não multiplos fluxos de caixa.
- O usuário deve ser capaz de realizar análises de investimentos mais apuradas através do cálculo do valor presente líquido (NPV), bem como da taxa interna de retorno (IRR).
- 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 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 de nossa calculadora deve ser o mesmo encontrado quando faz-se uso das calculadoras tradicionais.
- **Volume de Utilização**. A calculadora será mono-usuário, 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 em nossa calculadora não devam diferir 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").
- **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.

kk

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 [de Almeida 2006], conforme pode ser visto na figura 5.1:

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

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 estre os módulos se deu da forma mais simplificada possível para que não existisse maior sobrecarga que pudesse influenciar drasticamente o tempo de resposta desejados para as iterações com o usuário da aplicação, este foi um dos requisitos não-funcionais especificados na fase de análise. Outro

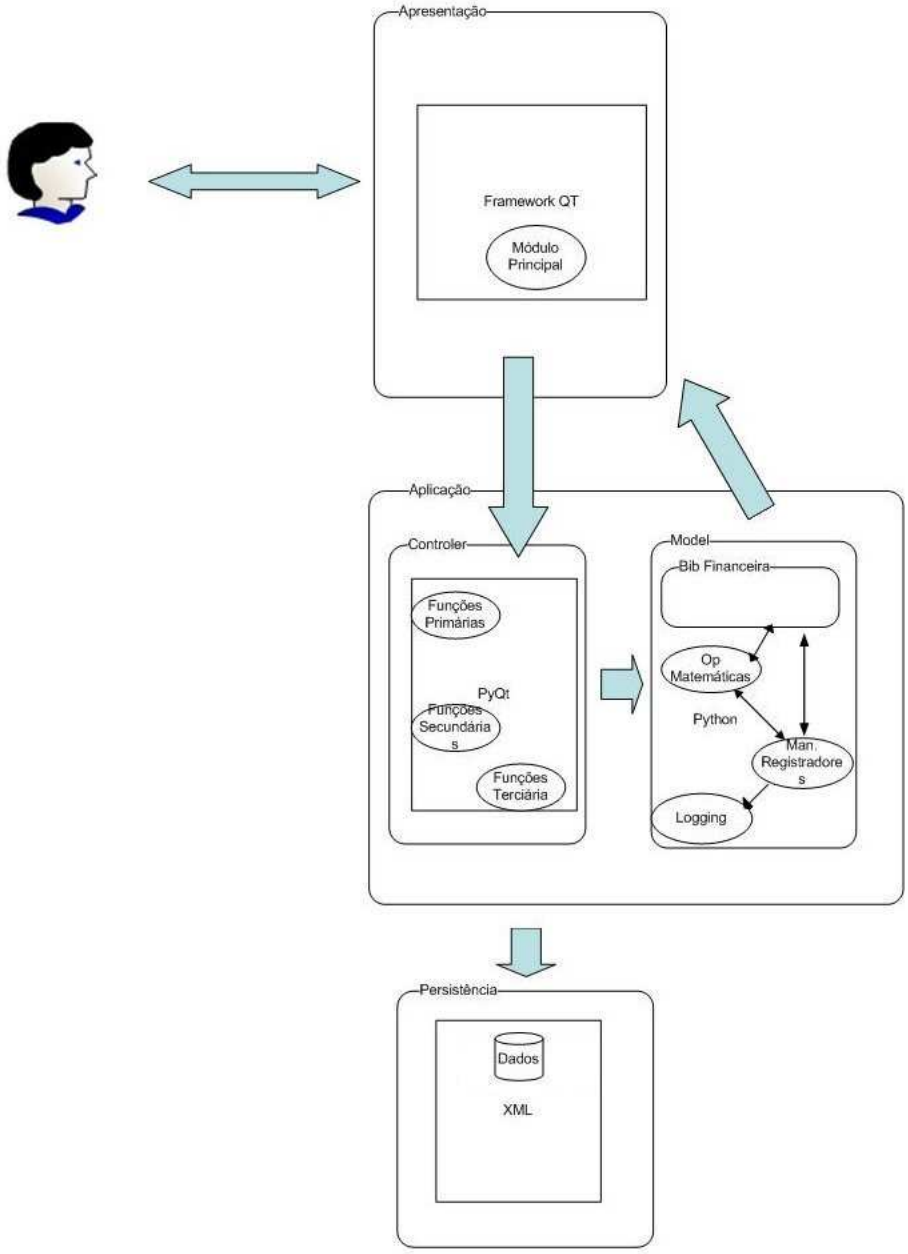

Figura 5.1: *Projeto arquitetural da aplicação.*

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

5.1.1 Camada de Apresentação

O sistema será acessado localmente por um único usuário via uma interface similar a da calculadora financeira que estamos usando como base, tendo a adição de menus de auxílio

para uso do sistema e uma opção de finalização da aplicação. É importante destacar ainda que toda entrada realizada pelo usuário se dará através da interface touch-screen disponibilizada pelo dispositivo selecionado. Para o desenvolvimento dessa camada faremos 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 de nosso sistema e foi desenvolvida em Python devido a suas vantagens na manipulação de números. Dentre os principais componentes desta camada podemos dar destaque à biblioteca financeira (também desenvolvida por esta 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 podemos destacar que as interações ocorrem em duas etapas:

1. Numa primeira etapa teremos 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 funcionalidade providas por cada tecla.

2. Numa segunda etapa teremos:

    - – 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. Usar o padrão XML para armazenamento nos pareceu ser uma solução viável de modo a evitar problemas com versões do Python que não contenham certos módulos relacionados à persistência de objetos. Os dados que forem armazenados nos arquivos deverão ser recarregados sempre que houver a inicialização da aplicação, demandando, assim, pouca interação com o disco nesse sentido. Outro aspecto relevante a ser considerada aqui é a segurança do sistema. Devido à natureza do sistema focamos 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 *Packing*

Foi tomada a decisão de distribuir a aplicação através de um arquivo de instalação de extensão .deb, simplificanco assim a maneira de intalação do software. Tal arquivo poderá ser obtido por download de certo servidor web que será escolhido pela equipe.

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 nosso desenvolvimento foi guiado pelo padrão de operações que é usado pela HP12-C que é a Notação Polonesa Reversa, ou Inversa segundo alguns autores [NPR]

Capítulo 6

Verificação e Validação

dsdsd

Capítulo 7

Métricas

sdsd

Capítulo 8

Conclusões

8.1 teste

asassd ....

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 empresa 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, insidindo 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, temos 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 datapólo 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 freqüência 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 as fontes a partir das quais as mesmas foram obtidas:

1. pv - BEG [Lutus a]:

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

2. pv - END [Lutus a]:

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

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

$$pv = fv + n * pmt$$

4. fv - BEG [Lutus a]:

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

5. fv - END [Lutus a]:

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

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

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

7. n - BEG [Lutus a]:

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

8. n - END [Lutus a]:

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

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

   - Se pólos com sinal igual:

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

   - c.c:

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

10. pmt - BEG [Lutus a]:

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

11. pmt - END [Lutus a]:

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

12. pmt com i = 0 [Lutus b]:

    - 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 b]

14. npv [HP 2009a]:

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

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]:

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

25. Variação Percentual [HP 2009a]:

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

26. Percentagem do Total [HP 2009a]:

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

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 acrescer 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 $1e-8$ 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 elabora-se 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.0.1 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 funcão 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 do modelo, da interfaçe gráfica e do controlador foi utilizado o modelo MVC [de Almeida 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 formatacão presentes no módulo calcFormatUtil.

C.0.2 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 os 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).

Uma implementação 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()
        else:
            # "inv /"
            try:
                result = self.__calcStack.getXReg() / self.__calcStack.
                    getYReg()
                self.__calcStack.rollCounterClockWise()
                self.__calcStack[0] = result
                self.__calcStack[3] = self.__calcStack[2]

                self.mode = Mode.OperationMode
                self.base = None
                self.dotActived = False

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

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

Bibliografia

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

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

[de Almeida 2006] de Almeida, R. R. (2006). Model-view-controller (mvc). [Online; acessado em 11-Maio-2009].

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

[eclipse] eclipse. [Online; acessado em 01-Abril-2009].

[Educação 2009] Educação, V. (2009). Curso planejamento financeiro pessoal. [Online; acessado em 24-Março-2009].

[FarsightSoft.com] FarsightSoft.com. [Online; acessado em 25-Março-2009].

[Fors] Fors, T. [Online; acessado em 25-Março-2009].

[G1 2009] G1 (2009). Saiba o que mudou no celular, o aparelho mais usado no brasil. [Online; acessado em 24-Março-2009].

[garage] garage. [Online; acessado em 01-Abril-2009].

[Gondim 2009] Gondim, M. C. L. (2009). Juros compostos e taxas equivalentes. [Online; acessado em 02-Maio-2009].

[HP] HP. [Online; acessado em 30-Março-2009].

[HP 2009a] HP (2009a). Hp 12c financial calculator user's guide. [Online; acessado em 24-Março-2009].

[HP 2009b] HP (2009b). Hp 12c internal rate of return. [Online; acessado em 24-Março-2009].

[HP 2009c] HP (2009c). Hp 12c net present value. [Online; acessado em 24-Março-2009].

[HP 2009d] HP (2009d). Hp 12c platinum calculadora financeira manual do usuário. [Online; acessado em 24-Março-2009].

[ISO] ISO. [Online; acessado em 01-Abril-2009].

[Lutus a] Lutus, P. [Online; acessado em 25-Março-2009].

[Lutus b] Lutus, P. [Online; acessado em 25-Março-2009].

[Mac 2009] Mac (2009). [Online; acessado em 24-Março-2009].

[Maemo a] Maemo. [Online; acessado em 25-Março-2009].

[Maemo b] Maemo. [Online; acessado em 25-Março-2009].

[Nokia a] Nokia. [Online; acessado em 25-Março-2009].

[Nokia b] Nokia. [Online; acessado em 25-Março-2009].

[NPR] NPR. [Online; acessado em 01-Abril-2009].

[Org] Org, M. [Online; acessado em 25-Março-2009].

[Pfützenreuter] Pfützenreuter, E. [Online; acessado em 25-Março-2009].

[pydev] pydev. [Online; acessado em 01-Abril-2009].

[pyeasyaccept] pyeasyaccept. [Online; accessed 01-Abril-2009].

[python.org] python.org. [Online; acessado em 25-Março-2009].

[pyunit] pyunit. [Online; acessado em 01-Abril-2009].

[Rangel 2009] Rangel, E. (2009). Resolução numérica de equações. [Online; acessado em 11-Maio-2009].

[Silva 2009] Silva, M. A. M. L. D. (2009). Módulo i - mat. financ. na hp-12c. [Online; acessado em 24-Março-2009].

[SVN] SVN. [Online; acessado em 01-Abril-2009].

[Wikipedia a] Wikipedia. [Online; acessado em 02-Maio-2009].

[Wikipedia b] Wikipedia. [Online; acessado em 02-Maio-2009].

[Wikipedia c] Wikipedia. [Online; acessado em 01-Maio-2009].

[XP1] XP1. [Online; acessado em 01-Abril-2009].