Projeto MyMon3y

Jaindson Valentim[1], **Matheus Gaudencio**[1]

[1]DSC – Universidade Federal de Campina Grande (UFCG)
Av. Aprígio Veloso, 882 – Bl. CN – Campina Grande – PB – Brazil

{jaindsonvs, matheusgr}@gmail.com

Resumo. *Este relatório técnico descreve o processo de desenvolvimento, a arquitetura e as tecnologias de apoio utilizadas na construção do projeto da disciplina "'Projeto de Software Orientado a Objeto" oferecida no período 2009.1 e ministrada pelo professor Dr. Jacques Philippe Sauvé.*

1. Introdução

Como parte das atividades da disciplina "Projeto de Software Orientado a Objeto" oferecida no período 2009.1 e ministrada pelo professor Dr. Jacques Philippe Sauvé, pediu-se um sistema de controle de finanças pessoais, o *MyMoney* (implementado com o nome de *MyMon3y* pelos desenvolvedores). Este relatório descreve a construção dos dois primeiros *milestones* do projeto: a definição das *User-Stories* e a codificação da lógica de negócio.

Este documento está estruturado da seguinte forma: a seção 2 apresenta a especificação do projeto, descrevendo as *User-Stories* planejadas e o modelo lógico de dados. Em seguida, a seção 3 mostra a arquitetura do software. Por fim, na seção 4, são descritas as tecnologias utilizadas na implementação do sistema.

2. Especificação

A especificação inicial foi fornecida pelo cliente do sistema (professor Dr. Jacques Philippe Sauvé) através do seguinte parágrafo:

> MyMoney é um sistema de controle de finanças pessoais onde **gastos e entradas** de dinheiro podem ser **registrados e classificados** e **relatórios** podem ser **gerados**. Há um **mecanismo de notificação de contas a pagar** e **mecanismos de importação de dados** de arquivos gerados como extratos do banco.

A marcação em negrito foi realizada pelos desenvolvedores deste sistema, e denota os elementos mais importantes extraídos da especificação. Tais elementos norteiam a definição das funcionalidades gerais do sistema:

- Registro de gastos e entradas de dinheiro (transações)
- Registro das transações por categorias
- Impressão de relatórios
- Notificação de pagamentos
- Importação de dados de extratos bancários

A partir da especificação, são definidos: o modelo lógico de dados (subseção 2.1) e as *User-Stories* (subseção 2.2).

2.1. Modelo Lógico de Dados

Foram definidas como entidades do modelo lógico de dados:

Usuário Representa a conta de um usuário com os atributos *login* (o email do usuário) e *senha*. Cada usuário dispõe de diversas *categorias* onde *transações* podem ser cadastradas.

Categoria Uma *categoria* agrupa transações de um determinado usuário.

Transação *Transação* representa uma movimentação financeira realizada pelo usuário. Apresenta uma breve *descrição*, a *data*, *comentário* e o *valor* da transação. Uma movimentação financeira de gasto ou entrada é indentificada pelo atributo *booleano crédito*, e há um atributo que descreve a *data de aviso prévio*, ou seja, a data em que uma notificação deve ser enviada ao usuário.

Estas são as entidades persistidas em que o sistema opera toda lógica de negócio. A apresentação da composição das entidades pode ser vista na figura 2.1.

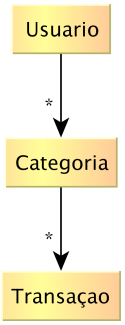

Figura 1. Modelo lógico de dados

O sistema ainda faz uso de uma entidade não persistente: **relatório**. Cada **relatório** representa uma agregação de **transações** de determinado **usuário** e existe apenas durante a operação de emissão de relatório.

2.2. User-Stories

Usando a especificação e as mensagens trocadas em lista e em avisos em sala-de-aula, foram definidas 8 **User-Stories** descritas à seguir. As **User-Stories** que trabalham alterando elementos do modelo lógico de dados apresentam também uma **User-Story** responsável pela funcionalidade da persistência de tais alterações.

Cada **User-Story** é especificada através dos seguintes elementos:

Descrição Pequena descrição do caso de uso.

Entidade Entidades do modelo de dados que são criadas, atualizadas ou removidas pela *User-Story*.

Persistência Determina se há ou não necessidade do tratamento de persistência das alterações realizadas no modelo de dados.

2.2.1. User Story 1 - Criação/edição de Conta

Descrição Criar uma conta de usuário no MyMon3y. Deve ser fornecido um email, que servirá de login, e uma senha.
Entidade Usuário.
Persistência Sim.

2.2.2. User Story 2 - Criação/edição de Categorias

Descrição Permitir a um usuário criar uma Categoria, fornecendo um nome. Não deve ser permitida a inserção de categorias com mesmo nome (case insensitive).
Entidade Usuário, Categoria.
Persistência Sim.

2.2.3. User Story 3 - Criação/edição/remoção de uma Transação

Descrição Permitir a inclusão/edição/remoção de uma Transação.
Entidade Categoria, Transação.
Persistência Sim.

2.2.4. User Story 4 - Remoção de Categoria

Descrição Remover Categorias que não possuem Transação associada.
Entidade Categoria.
Persistência Sim.

2.2.5. User Story 5 - Remoção de Conta

Descrição Remover Usuário do sistema.
Entidade Usuário, Categoria, Transação.
Persistência Sim.

2.2.6. User Story 6 - Geração de Relatórios

Descrição Gerar um relatório despesas de um determinado intervalo de tempo, organizando por Categoria e Transação para um Usuário.
Entidade Relatório.
Persistência Não.

2.2.7. User Story 7 - Importação de Dados

Descrição Importar dados de extratos bancários no formato OFX.
Entidade Transação.
Persistência Sim.

2.2.8. User Story 8 - Notificação de Transação

Descrição Verificar as Transações que devem ser notificadas pelo sistema num determinado dia. As notificações são enviadas diariamente.
Entidade -
Persistência Não.

3. Arquitetura

O sistema esstá estruturado através dos seguintes pacotes (diagrama de classes com os pacotes representado na figura 3):

com.google.code.mymon3y Primeiro pacote na hierarquia do projeto contendo a Facade do modelo de dados do sistema.
com.google.code.mymon3y.model Entidades do modelo de dados.
com.google.code.mymon3y.persistencia Classes responsáveis pela persistência.
com.google.code.mymon3y.util Leitor de OFX e classe para cálculo de hash.

Com este projeto, a classe **SistemaMymon3y** é responsável por atuar como *Facade* para a lógica de negócios. Por sua vez, **GerenciadorDePersistencia**, inicializada pela *Facade*, assume a responsabilidade por gerenciar a persistência dos dados. O sistema faz uso de classes utilitárias únicas do pacote modelo (no pacote util) e toda a validação dos dados são realizadas nas entidades do *model* e propagadas para os objetos que as utilizarem.

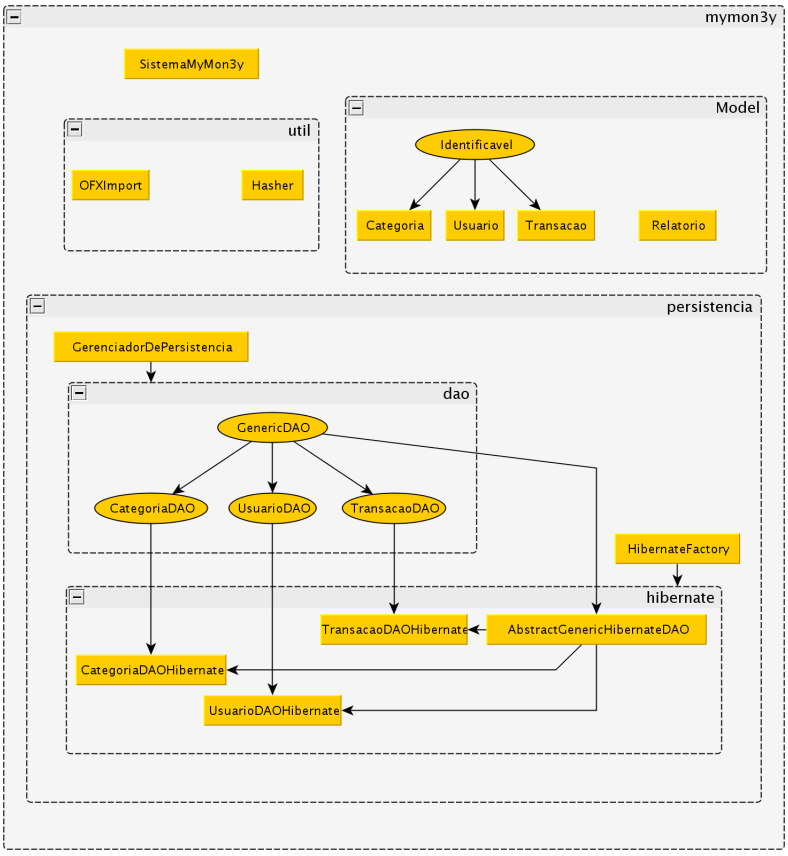

Figura 2. Diagrama de Classes

A *Facade* será utilizada em uma etapa posterior do projeto, estruturando o sistema através da arquitetura *Model-View-Controller*[**?**]. Nesta arquitetura, é clara a separação de uma lógica distinta para três entidades:

Model Implementa a lógica de negócio.
View Trata da visualização da interface com o usuário.
Controller Intermeia os pedidos oriundos da interface com o *Model*; manipula o estado da sessão; e em alguns casos, prepara o Model para ser entregue a *View*.

4. Tecnologias e Implementação

O projeto é desenvolvido em **Java** (1.6) utilizando o **Eclipse** como ambiente de desenvolvimento, o **SVN** oferecido pelo Google Code (`http://code.google.com/p/mymon3y`) como controle de versão e o **Ant** (1.7) como ferramenta de controle de construção do software.

Para dar suporte ao uso de testes de aceitação, utiliza-se a biblioteca **EasyAccept**. Para a persistência das entidades de dados, utiliza-se o **Hibernate** (3.0), que realiza o mapeamento de objetos Java a um banco de dados. O **Hibernate** é capaz de realizar tal mapeamento para diversos banco de dados, o sistema faz uso do **Apache Derby** (10.4) como o banco de dados utilizado. Esta escolha é motivada pela capacidade de funcionamento embarcado desta base de dados, permitindo que o ambiente de testes seja facilmente criado sem a necessidade de gerenciar ou ativar um serviço externo.