Universidade Federal de Campina Grande
Centro de Ciências e Tecnologia
Departamento de Sistemas e Computação
Disciplina: Laboratório de Programação II
Professora: Raquel Lopes

Relatório de Projeto

Central de Projetos da Computação@UFCG (CPC@UFCG)

Equipe:
Diego Eizi Takei Neto
Gilvan Ramos da Cruz Neto
Hugo Vinicius Araújo Fernandes
Jailson Barros da Silva Junior

Campina Grande - PB

Introdução

A Unidade Acadêmica de Sistemas e Computação (UASC) da Universidade Federal de Campina Grande (UFCG) é notoriamente conhecida pela excelência em suas atividades baseadas nos três pilares fundamentais de uma Universidade (Ensino, Pesquisa e Extensão). Com a finalidade de auxiliar a UASC no que diz respeito ao controle de Projetos, foi proposto a criação de um sistema que ajudasse na gerência dessa importante atividade desenvolvida pela Unidade.

Esse sistema, em resumo, teria diversas funcionalidades, dentre elas: Cadastrar Projetos e Pessoas, Remover Projetos e Pessoas, Editar Projetos, Associar Pessoas a Projetos, Calcular pontuação de uma Pessoa (de acordo com suas participações), Calcular valor das Bolsas de uma pessoa inclusa em um projeto (de acordo com a função do Participante), Gerência Financeira de Projetos e Geração de relatórios. A partir da implementação disponibilizada se deu início ao desenvolvimento do projeto.

Sobre o Design

No que se diz respeito ao design, foram utilizados os conceitos estudados ao longo das disciplinas de Programação II e Laboratório de Programação II: padrão de Desenvolvimento GRASP (*General Responsibility Assignment Software Patterns*) e MVC (*Model-view-controller*), e definições que englobam esses padrões: Encapsulamento, Ocultação da Informação, Acoplamento, Coesão, dentre outros.

O sistema pôde ser separado em diferentes módulos:

- ***Model's*** **de Participação, Pessoa e Projeto** - Continha as classes que definiam cada tipo exposto anteriormente além das demais classes que herdariam desses 3 tipos.

-***Factory*** - Tem como responsabilidade a criação de Pessoas e Projetos.

-***Controller*** - Composto basicamente das classes que tinham a responsabilidade de controle: *PessoasController* e *ProjetosController* que controlavam Pessoas e Projetos, respectivamente, e *CentralController* que englobava as duas classes apresentadas anteriormente.

-***Exceptions*** – Módulo composto pelas classes responsáveis pelo tratamento de exceções do sistema. Vale salientar que esse módulo continha implementações adicionais a cada novo caso de uso.

-***JUnitTestClass*** - Composto pelas classes de testes.

-***Facade*** - Onde estava localizada a Fachada do sistema.

Caso de Uso 1

O primeiro caso de uso tinha como objetivo adicionar, recuperar, editar ou apagar pessoas que já participaram ou irão participar de projetos da Unidade. Para isto, foi criada a classe *Pessoa* que representava uma pessoa com seus respectivos atributos e métodos. Pessoas seriam identificadas por um CPF e além disso possuíam um nome e e-mail. Foi criada também a classe *FactoryDePessoa,* que criava objetos do tipo pessoa.

Para adicionar, recuperar, editar ou apagar essas pessoas, foi criada a classe *PessoasController*, responsável por toda lógica destas funcionalidades. Nesta classe estão contidos os métodos: *cadastraPessoa, removePessoa, editaPessoa, getInfoPessoa*.

Caso de Uso 2

O caso de uso 2 consiste em adicionar, recuperar, editar ou apagar projetos. Foram criadas nessa parte as classes *Projeto* (que representava um projeto), *PED, PET, Monitoria e Extensão* (classes que herdam diversas características *Projeto*). Além disso, foi criada também a *FactoryDeProjeto*, que tem como responsabilidade a criação de objetos desse tipo. Foi necessário também uma classe que definia as despesas de um projeto, intitulada de *Despesas*.

Para operar toda a lógica pedida neste caso de uso foi implementada a classe *ProjetosController* contendo, inicialmente, os seguintes métodos: *adicionaMonitoria, adicionaExtensao, adicionaPED, adicionaPET, removeProjeto, editaProjeto, getInfoProjeto*, e os métodos get´s e set´s necessários.

Caso de Uso 3

O terceiro caso de uso tinha como funcionalidades adicionar, remover e pesquisar participações de pessoas em projetos. Foi criada neste passo a classe *Participacão* e algumas classes que herdariam algumas de suas características (*GraduandoParticipacao, PosGraduandoParticipacao, ProfessorParticipacao e ProfissionalParticipacao*).

Para gerir a lógica das Participações, dentre outras coisas, foi criada a classe *CentralController* que era composta inicialmente pelos métodos: *associaProfessor, associaProfissional, associaGraduando, associaPosGraduando, removeParticipacao, getPessoaParticipacao*, além dos métodos get´s e set´s. Vale salientar que assim como em todos os casos de uso, neste também foram implementadas classes que tratariam as exceções para cada situação.

Caso de Uso 4

O caso de uso 4 pedia o cálculo dos pontos de participação que uma pessoa já acumulou durante sua permanência em projetos. Para isto, foi implementada uma interface intitulada de *Pontuação* que continha o método abstrato calculaPontos e era implementada pela classe Participação e, obrigatoriamente (já que se trata de um método abstrato), por todas suas classes filhas (*GraduandoParticipacao, ProfissionalParticipacao, ProfessorParticipacao, PosGraduandoParticipacao*). Este design foi necessário pois cada tipo de Participação tinha um cálculo de pontuação diferente.

Caso de Uso 5

O caso de uso 5 consiste em calcular o valor da bolsa dos participantes dos projetos. Para isto, foi implementada uma interface intitulada de *ValorBolsa* que continha o método *calculaBolsa* e era implementada pela classe *Participação* e também por todas suas classes filhas (*GraduandoParticipacao, PosGraduandoParticipacao, ProfissionalParticipacao, ProfessorParticipacao*). Assim como no caso de uso anterior, este design é justificável pois cada tipo de Participação tinha um cálculo de bolsas distinto.

Caso de Uso 6

O caso de uso 6 serviria para gerenciar os recursos financeiros da UASC provenientes da colaboração dos projetos. Isso incluiria consultar e atualizar a receita da unidade (quando houvessem gastos dessa receita) e consultaria os projetos quanto à disponibilidade para atender demandas extras da unidade. Utilizando o princípio do Expert da Informação, foram criados os métodos *atualizaDespesas, calculaColaboracaoUASC* em *ProjetosController* e *calculaColaboracaoTotalUASC* em *CentralController,* que seriam responsáveis justamente pela funcionalidade destes casos de uso.

Caso de Uso 7

O caso de uso 7 consiste em gerar um relatório sobre o sistema, a respeito dos projetos cadastrados, do histórico de colaborações e do estado do caixa da unidade acadêmica. Para isso foi criada a classe *ProjetoPersistencia* que seria responsável pelo gerenciamento dos relatórios a respeito dos projetos. Neste design o controller de Projetos se comunica com esta

classe para que ela gere os arquivos e os salve externamente, a partir de um caminho relativo.

Caso de Uso 8

O caso de uso 8 tinha como objetivo a persistência de todos os dados do programa. Deveria ser salvos ao finalizar o programa e recuperados ao inicializá-lo. Para isso foi utilizado o conceito de fluxo (*Stream)* em Java nos métodos *iniciaSistema* e *fechaSistema* na classe *ProjetosFacade* para salvar os objetos do sistema em um arquivo externo, e também para recuperá-los para serem usado no sistema.

Link do Repositório: https://github.com/hugovaraujo/CentralDeProjetos.git