**Relatório de Projeto LP2 2019.2: Psquiza**

**Design geral:**

O design geral do projeto teve como principal objetivo manter o mínimo de acoplamento possível, para isso, optamos por não utilizar um Controller Geral e manter a Facade se comunicando diretamente com Controllers específicos de cada classe e com seus repositórios. O uso de repositórios foi escolhido a fim de manter uma alta coesão nas classes, dessa forma que o Controller específico de cada classe se preocupa apenas com a parte lógica da operação, enquanto o repositório fica responsável pelo armazenamento do objeto. Quanto ao uso dos Controllers específicos, optamos por utilizar-los para uma maior abstração do sistema. Quando a classe Busca foi implementada, percebemos que o baixo acoplamento havia sido comprometido, em discussão decidimos alterar a maneira como foi implementada, adotando a ideia de que tal classe deveria receber os repositório como parâmetro e percorrê-los sempre que um método for acionado, não havendo a necessidade de criar atributos na classe e mantendo sua associação apenas com a Facade. Quanto a organização de diretórios, foi criado um pacote para cada classe contendo além de si, seus semelhantes (Controllers específicos, classes filhas, etc), pacotes para os casos de teste específicos de cada classe e um pacote para os repositórios. Abaixo está detalhada a implementação de cada caso.

**Caso 1:**

**Responsável: Felipe de Souza Siqueira / Victor Hugo Sousa**

O caso 1 pede que seja criado um sistema de cadastro para as pesquisas, que contém uma descrição textual e campos de interesse relacionados a ela. Para isso, optamos por criar 3 classes relacionadas ao problema, a classe ControllerPesquisa, que contém a parte lógica relacionada ao cadastramento, alteração, encerramento, ativação, exibição e consulta do status de uma pesquisa, a classe Pesquisa, que representa o objeto pesquisa em si, e um repositório contendo todos os objetos Pesquisa. No atributo “campoInteresse” da classe Pesquisa, optou-se por utilizar um Array de Strings, visto que o parâmetro para tal

era passado como uma String única que deveria se separar e ser concatenada novamente em diferentes casos.

**Caso 2:**
**Responsável: Davi Andrade Pontes**

A US2 exige a criação de uma classe pesquisador, com uma função, nome, biografia, e-mail e um link para uma foto. o Pesquisador deve ser identificado pelo seu e-mail. Escolhi fazer inicialmente o pesquisador de forma simples, uma classe com os atributos de email, nome, função, biografia e foto, todos do tipo String, sendo o atributo foto um link para uma foto do pesquisador.

Além da classe de pesquisadores foi feita uma classe que armazena os pesquisadores; o repositório de pesquisadores. O repositório guarda todos os pesquisadores cadastrados no sistema Psquiza. A classe de repositório armazena os pesquisadores e somente realiza operações básicas sobre eles, sem ter acesso às funções do pesquisador. Somente manipula os pesquisadores sem interagir com eles. Toda interação com os pesquisadores é feita pelo controller de pesquisadores.

**Caso 3:**
**Responsável:**

**Caso 4:**
**Responsável: Victor Hugo Sousa**

A US4 pede que seja criado um sistema para o controle de atividades, incluindo as seguintes funções : cadastrar, apagar, e exibir atividades, além de cadastrar e contar itens que podem ser cadastrados na atividade. A classe controllerAtividades vai obter o acesso ao dados passados pelo usuário, processá-los e passar as informações necessárias para a classe repositorioAtividades, que tem a função de armazenar, excluir, verificar a existência e retornar atividades.
Ademais, foi utilizado o objeto ArrayList como forma de armazenamento de ids, pois é pedido ordem na função de listar atividades, e hashmap para um acesso mais eficiente nos outros métodos, A classe Atividade

também se mostrou necessária, visto que toda atividade tem descricao, nível de risco e descrição de nível de risco, e pode ter itens associados a ela ou não, além de tudo foi criada a classe IDatividade já que a chave do hashmap é gerado automaticamente.

**Caso 5:**
**Responsável:**

**Caso 6:**
**Responsável:**

**Caso 7:**
**Responsável: Davi Andrade Pontes**

Para a US7 foi preciso modificar a classe de atividades e de pesquisas, para que ambas pudessem ser associadas e foi criado um método para isso; foi usada composição, pois a pesquisa passou a ter uma coleção de atividades que a compõem e a atividades passou a conhecer a identificação das pesquisas as quais ela pertence.

**Caso 8:**
**Responsável: Felipe de Souza Siqueira**

Na US8 o objetivo é que seja oferecida ao usuário uma maneira de percorrer certos atributos das classes a procura de uma palavra específica, dessa forma foram implementados 3 métodos (busca, contaResultadosBusca e busca) e a classe Busca. Aqui considerei necessário sobrecarregar o método busca, visto que o usuário poderia buscar por todas as frases que tivessem a palavra escolhida ou acessar diretamente alguma a partir de sua posição. Após discussões com o grupo, chegou-se à conclusão de que a classe Busca deveria receber os repositórios como parâmetros, pois assim seria possível manter todas as operações de uma busca dentro de uma única classe, mantendo um baixo acoplamento e aumentando a coesão do sistema. Assim, a classe Busca ficou responsável por varrer os repositórios das classes Atividade, Objetivo, Pesquisadores, Pesquisas e Problemas e

resgatar (através de métodos get) atributos com a palavra desejada, concatená-los, ordená-los e/ou contá-los.

**Caso 9:**
**Responsável:**

**Caso 10:**
**Responsável: Davi Andrade Pontes**

Na US10 foi usado o padrão de design Strategy, para ordenar as atividades de uma pesquisa dado um certo critério definido pelo usuário em tempo de execução. Após essa ordenação, como exigido pela especificação, o Psquiza pega a atividade que, de acordo com o critério definido, é recomendável que seja executada em seguida pelo usuário.

**Caso 11:**
**Responsável: Felipe de Souza Siqueira**

No caso 11, a especificação indica que uma pesquisa, os pesquisadores envolvidos nela, o problema, os objetivos e as atividades desta pesquisa sejam exportados para um arquivo de formato txt. Além disso, deve ser possível também gerar um arquivo com os resultados obtidos em uma pesquisa. Para a resolução do problema, dois métodos foram criados na classe ControllerPesquisa, foram eles “gravarResultados()” e “gravarResumo()”, ambos recebem o ID da pesquisa como parâmetro e constroem a String que será gravada, dentro de si mesmos. Para a adequação ao problema, umas das estruturas de dados da classe Pesquisa foi trocada de HashMap, para LinkedHashMap, visto que os pesquisadores precisavam estar gravados na ordem em que foram criados.

**Caso 12:**
**Responsável:**