Relatório de Projeto LP2 XX

Design geral:

O design geral do nosso projeto foi escolhido para possibilitar uma melhor integração entre as diferentes partes do sistema, criando camadas de abstração que diminuem o acoplamento, através do uso de um controller geral, que delega atividades para cada camadas menores. As camadas menores também possuem um gerenciador próprio, para aumentar o nível de abstração do sistema.

Para a criação de novos objetos, adotamos o uso de factories. O uso de enumerators foi

pensando para o uso em situações onde o atributo seria imutável, facilitando a catalogação no sistema.

Quando uma entidade precisou de comportamento dinâmico, escolhemos o padrão strategy . Com relação a exceptions , fizemos uma hierarquia que leva em consideração o tipo de erro.

Logo, temos exceptions para atributos inválidos, tanto de criação como de atualização de dados, para erros de sessão e de permissão. Todas estão encapsuladas em packages menores, e herdam de uma mais geral do projeto: ProjetoEX .

As próximas seções detalham a implementação em cada caso.

Caso 1:

O caso de uso 1 pede que seja criado uma entidade que represente uma pesquisa no sistema, essa pesquisa possui um código de identificação único, descrição e campo de interesse. Para gerenciar as pesquisas foi criado um controller pesquisa, que armazenar a entidade através de um mapa que usa o código da pesquisa como chave, o código é gerado automaticamente ao cadastrar a pesquisa considerando os outros códigos para não criar um igual. Também foi solicitado que seus atributos pudessem ser alterados, exceto o código da pesquisa. A pesquisa deve ser ativada ou desativada e para isso foi adicionado um atributo booleano que indique o estado de ativação dela. Seu método de exibição é usado através da sua representação em String.

Caso 4:

O caso de uso 4 pede para que sejam criadas atividades, cada atividade no sistema terá uma descrição, um nível de risco, podendo ser alto, médio ou baixo, e a descrição deste risco. Além disso, uma atividade poderá ter itens a serem realizados, para armazenar os itens, a atividade terá uma lista de itens da classe Item. Esta classe irá encapsular o nome e status do item. Todas atividades serão gerenciadas por um controller, chamado ControllerAtividade, e este fará todas as operações relacionadas à atividade, como cadastrar, apagar, exibir, incluindo as operações de itens da atividade, como cadastro e contador de itens por status.

Caso 5:

O caso de uso 5 pede para que seja criada uma associação de pesquisa com problema e objetivos, cada pesquisa só poderá ter um problema associado e vários objetivos associados. Um problema poderá existir em várias pesquisas, porém um objetivo não poderá existir em diferentes pesquisas. Como pesquisa possui problema e objetivos, a associação foi feita na própria classe Pesquisa, onde um objeto de pesquisa possui Problema e um mapa de Objetivos como atributo. As operações de associação foram colocadas em um ControllerGeral, que faz a conexão entre os controllers do projeto. Ao associar um problema ou um objetivo a uma pesquisa, o controller geral usa o controller de problema e objetivo para retornar um objeto que será colocado na classe pesquisa. O controller de pesquisa fará as operações deste caso de uso, como associação e desassociação de problema e objetivos, e a listagem de pesquisas por ordem. Na listagem de pesquisa foi necessário o uso de comparators para ordenar as pesquisas por ID de problema, quantidade de objetivos e código da pesquisa. Para facilitar a ordenação em mapas foram usadas as operações referentes a stream, juntamente com o conceito de Lambda em java.

Caso 6:

O caso de uso 6 pede que seja associado e desassociado pesquisadores a uma pesquisa, para isso foi criado um mapa dentro de pesquisa que armazena pesquisadores e para desassociação de um pesquisador bastou removê-lo do mapa, pesquisadores também podem ser listados de acordo com seu tipo. Um pesquisador pode ser externo, aluno ou professor, sendo possível cadastrar detalhes apenas ao aluno e professor. Para representar essas especialidades que possuem diferentes características e ainda continua sendo um pesquisador foi criado uma interface que possui os métodos altera e toString, a interface foi implementada em duas novas classes criadas, professor e aluno, assim tornou-se possível a alteração de atributos de ambas as especialidades que representam seus detalhes e capturar sua representação em String.

Caso 11:

O caso de uso 11 pede que seja criado dois arquivos de textos, com o resumo e resultado de uma pesquisa. Para realizar a escrita das informações em um arquivo de texto(.txt) foi usado o controller de pesquisa para juntar todas as informações e colocar em uma String. Para escrever os dados em um arquivo foi usado o fluxo de dados de saída.

Caso 12:

O caso de uso 12 pede para que seja armazenado tudo que é cadastrado no sistema, no caso, todas as instâncias dos objetos que são criados de todas as entidades, como Pesquisa, Pesquisador, Atividade, Problema e Objetivo, e para que seja carregado este armazenamento no início da execução. Para salvar todas as informações cadastradas, todos os mapas de todos os controllers foram inseridos em um arquivo com nome “psquiza.txt”, para isso foi necessário que todas as entidades do sistema fizessem a implementação da interface *Serializable*, além da inserção de todas entidades no sistema, foi necessário gravar o contador que é utilizado para gerar novos códigos no cadastro de problemas, atividades e objetivos. Com todas as informações salvas, o carregamento foi feito de maneira similar, a partir do arquivo “psquiza.txt” foram carregados todos os dados na ordem que foram inseridos.

Considerações:

Optamos por fazer o projeto todo em inglês. Logo, o mesmo está propenso a pequenos erros de ortografia/gramática, já que não somos fluentes no idioma. Mesmo assim, nos esforçamos para

evitar deslizes. Apenas a Facade está em português, para que os métodos sejam compatíveis com as chamadas nos testes de aceitação.

Link para o repositório no Bit Bucket:

https://bitbucket.org/o0000/soossososos/