Relatório de Projeto LP2 2019.2
Sistema Psquiza

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 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 dos controllers. O uso de contadores dentro de controllers foi pensando para o uso em situações onde o atributo seria imutável, facilitando a catalogação no sistema.
Com relação a exceptions, fizemos exceções básicas para validação de entradas, tais como NullPointerException, IllegalArgumentException e RuntimeException.
Logo, temos exceptions para atributos inválidos, tanto de criação como de atualização de dados, para erros de sessão e de permissão.
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 2:
O caso de uso 2 pede que seja criada uma representação de um pesquisador no sistema, esse é identificado unicamente por um email, e possui também Nome, Função, Biografia e uma URL indicando a Foto. Para serem válidos, o email deve possui uma letra ou número seguido de uma arroba (“@”) e o URL de foto deve começar com “http://” ou “htttps://” seguido do endereço. O pesquisador pode estar ativo ou inativo, representado por um atributo boolean. Para gerenciar os pesquisadores, foi criado um Controller Pesquisador, que armazena pesquisadores em um Mapa, identificados pelo email. As informações do pesquisador podem ser alteradas, incluindo o email.

Caso 3:
O caso de uso 3 pede que sejam criadas entidades que representam objetivos e problemas no sistema. Os objetivos possuem tipo (que pode ser GERAL ou ESPECIFICO), descrição, aderência e viabilidade (os dois últimos sendo valores inteiros entre 1 e 5). Os problemas possuem apenas descrição e viabilidade. Ambos os tipos de entidade são gerenciados por controllers específicos: controller objetivo, para os objetivos e controller problema para os problemas. As entidades são armazenadas em mapas que têm como chave códigos gerados quando elas são cadastradas, O + um id para os objetivos e P + um id para os problemas. O id é um número que indica a ordem de cadastro das entidades. Também foi solicitado que fosse possível apagar e exibir uma representação em String de ambas as entidades.

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 do tipo 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 pendências.

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 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 sejam associados e desassociados 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 7:
O caso de uso 7 pede que sejam associados e desassociados atividades a uma pesquisa. A mesma atividade pode estar associada a diferentes pesquisas. A associação é feita ao armazenar a atividade num mapa dentro de pesquisa e para desassociar a atividade basta removê-la do mapa. Também foi pedido que fossem implementadas uma função que permita executar itens de uma atividade a partir de uma duração, mudando o seu status e uma função que retorne a duração total dos itens de uma atividade. Apenas atividades que estejam associadas a uma pesquisa podem ter seus itens executados. Além disso foi solicitado que fosse possível registrar resultados em uma atividade, que são representados em String e identificados por um índice. Para isso foi criada uma lista de resultados em atividade.

Caso 8:
O caso de uso 8 pede que seja criada uma funcionalidade que permita fazer buscas de um termo no sistema, essa busca é feito nos campos Descrição e Campo de Interesse em Pesquisa, Biografia em Pesquisador, Descrição em Problema, Descrição em Objetivo, Descrição e Descrição de Risco em Atividade. A busca retorna todos os campos onde o termo for encontrado totalmente ou parcialmente presente, desconsiderando a capitalização do termo e dos campos. Para realizar a busca é utilizado uma entidade Buscador, com acesso aos Controllers individuais, que possuem métodos responsáveis por fazer a busca em suas entidades específicas e ordenar os campos buscados, em seguida os resultados das buscas individuais são agrupados numa busca geral. As buscas feitas podem ser Geral: retornando todos os resultados encontrados ou por um resultado específico, determinado através da posição entre os resultados encontrados. Também é possível contabilizar a quantidade de resultados encontrados na busca.

Caso 9:
O caso de uso 9 pede que seja possível relacionar Atividades de modo a definir uma ordem de execução, em que uma Atividade pode ser definida como subsequente a outra. Para a implementação, foi criado um atributo em Atividade que indica a atividade subsequente, é possível definir uma relação de ordem através dos códigos identificadores das atividades, também é possível remover essa relação, quebrando a cadeia de execução. Também é possível contar a quantidade de atividades subsequentes a uma certa atividade, sendo feito uma contagem utilizando recursão até a última atividade da ordem. É possível ainda pegar o código da enésima atividade subsequente a uma atividade desejada, ou seja, retornar o código da enésima posição em relação a Atividade indicada e também pegar o código da atividade de maior risco em uma ordem de execução, a partir de determinada atividade.

Caso 10:
O caso de uso 10 pede que seja implementada uma função que retorna o código de uma atividade entre as associadas a uma determinada pesquisa que deve ser a próxima a ser executada. Apenas uma atividade que ainda possui itens pendentes é válida para ser retornada. A estratégia utilizada na seleção dessa atividade é definida por uma outra função e pode ser 4 estratégias distintas: Mais antiga, onde é retornado o código da atividade válida que foi associada àquela pesquisa a mais tempo; Menos pendências, onde é retornado o código da atividade válida que possui menos pendências; Maior risco, que retorna o código da atividade válida que possui o maior risco e Maior duração, que retorna o código da atividade válida que possui a maior duração total. Caso haja empate em alguma das últimas 3 estratégias o critério de desempate é qual delas é a mais antiga na pesquisa. Para isso foi criado um atributo String estratégia em controller pesquisa. O valor atual deste atributo é passado para a pesquisa ao ser executada a função de retornar a próxima atividade e a partir do seu valor é executado um determinado case em um switch case dentro de pesquisa, que sugere a atividade de acordo com a estratégia atualmente definida.

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:
-

Link para repositório no Github:
https://github.com/davibss/Psquiza