Universidade Federal
de Campina Grande

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CIÊNCIA DA COMPUTAÇÃO
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA - CEEI

IELE FACUNDO PASSOS
JOSÉ RICARDO GALDINO FELIX
KLEBERSON JOHN SANTOS DE MARIA
THIAGO DA COSTA LIRA

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE – UFCG

RELATÓRIO PROJETO LP2 – 2019.2

CAMPINA GRANDE – PB

2019

Relatório de Projeto LP2 – 2019.2

- Design Geral:
O design geral do nosso projeto foi elaborado de modo à permitir que distintas partes do sistema pudessem se integrar, criando camadas de abstração e diminuindo o acoplamento à partir de um controller geral que delegava responsabilidades às camadas menores. Cada uma dessas camadas é gerenciada por um controller específico que trabalha sobre determinados objetos.
Construindo o sistema com base no padrão Strategy, fomos cautelosos quanto aos experts e creators, delegando uma ação à quem realmente deveria fazê-la.
Quanto ao lançamento de exceções, criamos uma classe validadora para verificar se as entradas e operações são válidas ou não.
Ao longo das próximas seções, temos um melhor detalhamento de cada caso de uso e das soluções que propomos.

- Caso 1:
No caso de uso 1, houve a necessidade de criar um controller para pesquisa, juntamente com a classe Pesquisa. Uma pesquisa é identificada unicamente por seu código, por isso, o ControllerPesquisa tem um mapa tento a chave código do pesquisa e o valor o objeto Pesquisa, que, além de seu código, possui descrição e campo de interesse.

- Caso 2:
No caso de uso 2, percebeu-se a necessidade de criar um controller para pesquisador, juntamente com a classe Pesquisador. Um pesquisador é identificado unicamente por seu email, sendo assim se tornou viável o ControllerPesquisador ter um mapa tendo a chave o email e o valor o objeto Pesquisador, que possui alem de email, nome, funcao, biografia e url para foto.

- Caso 3:
Para o Caso 3, criamos dois novos controllers: um para trabalhar com Problema e outro com Objetivo. O ControllerProblema possui uma coleção (Map) de problemas, a classe Excecao para indicar erros, e gerencia operações básicas do sistema para com Problema, como cadastrar, exibir e apagar. O ControllerObjetivo segue o mesmo modelo. Ambos também possuem um atributo contCodigo, que é usado para gerar códigos de cada objeto.
Cada problema tem como atributos código, descrição e viabilidade, enquanto os objetivos têm código, tipo (se geral ou específico), descrição, aderência e viabilidade.

- Caso 4:
No caso de uso 4, que se refere às atividades e itens, optamos por criar um controller separado para atividades, com seus métodos e os de itens(não foi necessário um especifico para este), que por sua vez estão associados com atividade. A ordem dos itens é mantida pelo identificador único de cada item e a não repetição de itens por meio de uma coleção (Map), que é a estrutura que os guarda.

- Caso 5:
O Caso 5 explicitou a necessidade de trabalhar sobre mais de um controller. Com isso, decidimos implementar um controller geral. Para que as associações sejam realizadas é

necessário que a pesquisa em questão esteja ativada. Desse modo, uma pesquisa só pode estar associada à um único problema, mas pode estar associada à vários objetivos. Os objetivos, por sua vez, só podem estar associados à uma pesquisa, e os problemas podem estar associados à diversas pesquisas. Por conseguinte, uma pesquisa agora possui um problema e uma coleção (Map) de objetivos. Quando um problema é desassociado, o atributo problema em pesquisa, passa a ser null, e na desassociação de um objetivo, o objetivo passado é tirado da coleção.
Por fim, é possível listar as pesquisas, que pode acontecer por ordem de problemas (maior ID de problema ao menor), objetivos (quantidade de objetivos) e pelas pesquisas de maior ID.

- Caso 6:
Na parte da associação entre pesquisadores e pesquisas, foi necessário novos métodos e atributos, em especial as pesquisas que ganharam uma lista de pesquisadores. No tocante a especialização do pesquisador, caso seja aluno ou professor, esta é dada por meio de um objeto que guarda suas especialidades, que interagem com os pesquisadores por meio de interface. Foram necessárias modificações em métodos já implementados, em especial o toString, que dependia da especialidade.

- Caso 7:
O caso 7 mostrou a necessidade de mais um controller e, assim, optamos por implementar um controller geral (controllerPsquiza). A Us7 é responsável por associar e executar atividades e, para que isso aconteça , a pesquisa deve existir e estar ativa. Além disso, para executar um item duma atividade, deve-se garantir que ela exista , já esteja associada a uma pesquisa e o item não tenha sido previamente realizado.

- Caso 8:
Como a especificação de busca navega por todos os objetos do sistema, decidiu-se adotar um controller geral, uma classe que administra todos os controllers e por conseguinte todas as classes. Para as funcionalidades de busca, foi implementado um método privado no ControllerPsquiza que retorna uma lista com todos os resultados na ordem desejada, Este método invoca em cada controller (na ordem especificada) um metodo de busca que retorna uma lista de resultados (de classes Busca que contem o campo em que o termo foi achado e uma identificação), por exemplo, todos os pesquisadores que possuem o termo. A partir desse método foi fácil implementar uma busca simples, busca um resultado especifico (posição da lista) e a quantidade de resultados (tamanho da lista).

- Caso 9:
A grande mudança nesse caso foi a atividade sugerir qual seria sua "próxima" atividade, uma ordem própria sugerida. Para tal, criamos um atributo que mantinha essa informação, que pode ser alterado a qualquer momento, entretanto modifica a "ordem natural" das atividades. Utilizamos recursividade em pegaProximo e pegaMaiorRisco, por parecer melhor para o problema em questão, mantendo a organização e estruturação da ideia principal da especificação.

- Caso 10:
Para que o sistema escolhesse uma próxima atividade para ser realizada foi criado um atributo "estrategia" no ControllerAtividade que é instanciada inicialmente como "MAIS_ANTIGA". Para configurar um estratégia, o sistema verifica se a estratégia passada é válida e modifica a estratégia do sistema. Na necessidade de retornar a

próxima atividade de acordo com a estratégia definida foi implementado um método no ControllerAtividade que verifica qual estratégia está definida e para cada uma existe um método privado que busca, ordena as atividades pendentes e retorna a próxima atividade a ser realizada.

- Caso 11:
O Caso 11 nos apresentou um novo desafio: exportar os dados do nosso sistema. No entanto, o caso não foi de grande complexidade. O controller geral foi responsável por realizar a exportação do(s) resumo/resultados da pesquisa. Os resumos são gerados pelo objeto Pesquisa à partir, principalmente, dos métodos toString() de cada objeto. Os resultados são formatados em Pesquisa, e gerados em Atividade, já que cada resultado diz respeito à uma atividade associada à uma pesquisa.

- Caso 12:
O caso 12 apresentou a necessidade de salvar e ler os dados do sistema, de forma a garantir a permanência do estado da aplicação entre execuções.
Dessa forma, cada controller do sistema, com exceção do geral, possui um método responsável por salvar os dados em um arquivo. Assim, cada um é salvo num arquivo diferente.

- Considerações:
O projeto foi de suma importância para que pudéssemos desenvolver nossas habilidades de desenvolvimento de sistema e de nos relacionarmos em grupo, trabalhando juntos em prol de um mesmo objetivo. A complexidade do projeto e as delegações de determinados casos de uso à cada membro de nossa equipe, assim como termos que analisar os códigos uns dos outros foi um desafio interessante, que podemos superar com esforço e companheirismo.