­Relatório de Projeto LP2 – Matheus Silva Araújo; Gabriel Guimarães de Almeida.

Design Geral:

O design escolhido para o projeto foi estudado a fim de proporcionar uma melhor sistemática entre os componentes do mesmo. Na camada superior existe uma Fachada que tem por objetivo servir de ‘meio campo’ entre os comandos do usuário e o código mais inferior do sistema. Logo abaixo há um controller dos controllers o qual delega funções para os controllers inferiores e que contém certa lógica de implementação própria, a fim de proporcionar um menor acoplamento entre as partes.

Para a criação de novos objetos, adotamos o uso de factories. O uso de herança foi utilizado a fim de reutilizar um mesmo código para diversas classes filhas.

Com relação as exceções, é possível captura-las e tratá-las de acordo com a especificação do projeto. Ao discorrer deste presente relatório, mais detalhes serão evidenciados.

Caso 1:

No primeiro caso, pede-se a possibilidade de cadastrar itens no sistema, tais itens podem cadastrados de acordo com o seu tipo (por quantidade, por kg, e por unidade), para isto, escolhemos o reuso de código, ou seja, o uso de herança para fomentar as classes que estendem ‘Item’, o método ainda lança exceções caso haja alguma entrada inválida. Os Itens são armazenados numa coleção do tipo HashMap, onde o seu id é a chave, e o próprio item é o valor. O HashMap foi inicializado no ControllerItens, que por sua vez gerencia todos os métodos que condizem com Item. O método de listar os itens de acordo com o seu id (identificador único do item) foi implementado a fim de retornar o status do item cadastrado, com seu nome, unidade, preço e local de compra, caso haja algum id inválido, o método irá retornar uma exceção. Todo Item pode ter mais de um preço a partir de outros estabelecimentos, para isto, foi usado outra vez um HashMap, cujo a chave é o local de compra, e o valor é o próprio preço.

Caso 2:

No caso de uso 2, pede-se que os itens sejam listados a partir da ordem em que foram cadastrados no sistema, para isso foi criado o método getItem(int) que recebe a posição que o usuário deseja do item na lista. Não foi preciso criar novas entidades para este caso. Também é possível listar os itens a partir da sua categoria (getItemPorCategoria(str, int)), pelo preço crescente (getItemPorMenorPreco(int)), e pela pesquisa do usuário (getItemPorPesquisa(str)).

Caso 3:

No caso de uso 3, pede-se a criação de lista de compras, para isto, foi implementado um ControllerListas o qual armazena uma coleção de listas de compras através de um HashMap, cujo a chave é o nome da lista de compra (descritorLista),e a chave é a própria lista de compras. Nesse controller ainda é possível adicionar, atualizar, modificar e remover compras de uma lista de compras. Por fim, é possível finalizar uma lista de compras, informando o seu descritor e o valor final da compra, e imprimir de acordo com o seu status.

Caso 4:

Neste caso de uso, é possível recuperar listas de compras a partir das datas as quais foram cadastradas. Ainda é possível recuperar uma compra de acordo com a sua posição na lista, e o seu descritor. Nesse caso de uso, novas entidades não foram implementadas.

Caso 5:

Nesse caso de uso, pede-se a geração automática de listas de compras, para isso, foram implementados métodos de geração automática de listas por: última lista de compras feita (geraAutomaticaUltimaLista()), última lista cadastrada a qual contém o item que foi informado na entrada do método (geraAutomaticaItem()), e por fim, uma lista com os itens que aparecem em mais de 50% das listas já cadastradas no sistema (geraAutomaticaItensMaisPresentes()). Nesse caso, não foram criadas novas entidades.