



Universidade Federal  
de Campina Grande

**UNIVERSIDADE FEDERAL DE CAMPINA GRANDE**  
**DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO**  
**CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO**

**PROJETO DE LABORATÓRIO**  
**LABORATÓRIO DE PROGRAMAÇÃO 2**

CAMPINA GRANDE  
AGOSTO DE 2018

**PROJETO DE LABORATÓRIO**  
LABORATÓRIO DE PROGRAMAÇÃO 2

**Aplicativo**  
**ListaPraMim**

José Guilherme Coelho de Oliveira  
Mariana Silva Nascimento  
Thayanne Luiza Victor Landim Sousa  
Siuanny Barbosa dos Santos Rocha

**Professora**  
Eliane Araújo

**Monitor**  
Marcos Barros

# Padrões de projeto e de arquitetura utilizados

No projeto, foram utilizados vários padrões GRASP (General Responsibility Assignment Software Patterns), afim de tornar o projeto mais flexível e de melhor compreensão:

- **Creator:** Considerando que a criação de objetos é uma das bases da OO, o creator ajuda na organização da criação desses objetos, pois a criação é atribuída apenas às classes que fazem sentido.
- **Controller:** O Controller é de uso imprescindível em um projeto como o ListaPraMim, pois é necessário que haja um desacoplamento dos eventos gerados pela interface e dos objetos por tratar a requisição, deixando o sistema mais flexível. Inclusive, foi criado um Controller “mediador” de outros dois Controllers, para que a comunicação com o Facade ficasse mais limpa e organizada.
- **Alta Coesão:** Esse padrão foi utilizado para que os objetos pudessem ser mantidos de forma gerenciável e compreensível. Pois, o fator chave da alta coesão é que as responsabilidades de um determinado elemento devem ser mantidas de forma bem relacionada.
- **Baixo Acoplamento:** Alta coesão e baixo acoplamento são inversamente proporcionais. Se há uma alta coesão, espera-se que exista um baixo acoplamento. Pois, se as responsabilidades dos membros estiverem acontecendo de forma relacionada e focada (alta coesão) isso significa que os módulos (detentores dos membros) não serão tão dependentes entre si. Assim, as classes do projeto mantiveram apenas as relações necessárias de acordo com suas funcionalidades. Deixando o projeto mais flexível e facilitando nos ajustes necessários.

Também foi usado um padrão GoF (Gang of Four), que foi necessário na execução da leitura e geração de arquivos.

A classe Persistência, tem como responsabilidade leitura e recuperação de arquivos, salvando objetos utilizados ao longo do sistema. Nessa classe foi utilizado o padrão Singleton, que garante que a Persistência seja instanciada uma única vez. Sendo um ponto global e acessível para todas as classes do sistema.

# O que foi feito e como foi feito

Aqui iremos explicar a criação de algumas classes essenciais para cada caso de uso (não especificamente todas serão mostradas aqui).

- No caso de uso 1 (CRUD dos itens compráveis), foi estabelecida uma relação de herança, onde a classe pai é ItemCompravel e as classes filhas são ItemPorQtd, ItemPorUnidade e ItemPorQuilo. A relação de herança foi utilizada seguindo a lógica do “é um”, ou seja, um item por quilo *É UM* item comprável. Foi criado também o controller ItemCompravelController, uma classe Enum para as categorias dos itens e outra para as exceções (mensagens de exceção) que seriam lançadas.
- No caso de uso 2, foram criadas classes Comparador e implementados outros métodos no ItemCompravelController para que fosse possível a ordenação dos itens nos modelos que foram pedidos.
- No caso de uso 3, foram criadas as classes Compra, Lista de Compras, um controller para gerenciar as listas de compras e uma classe Comparador para ordenar a lista por categoria e ordem lexicográfica. A decisão de criar a classe Compra separadamente visou facilitar a criação e manipulação de listas de compras.
- No caso de uso 4, foram adicionados métodos nas classes ListaDeCompras e ListaDeComprasController para poder ser feita a pesquisa de listas por descrição e por data. Para adquirir a data atual, foi criado um método que captura a data do sistema e, em seguida, esta data é convertida para o formato desejado.
- No caso de uso 5, foram implementados apenas novos métodos para geração automática de listas de compras.
- No caso de uso 6, foi utilizada a operação merge (criada no Java 8). Essa operação dá merge de todos os mapas de preços dos itens das listas de compras, e, após isso, é criado um Map <String, double> em que a String representa o estabelecimento e o double é o valor total da lista de compras naquele estabelecimento. Depois, é feita uma ordenação com base no preço e os supermercados são passados para um TreeMap que oferece uma ordenação diferente do HashMap. São usadas as funções Stream() e Filter() para acessar os itens nos estabelecimentos que o usuário preferir. Por fim, foram criados métodos de pesquisa e adicionados ao ListaDeComprasController.
- No caso de uso 7, foi criada a classe Persistência, feita no padrão singleton. Essa classe é instanciada apenas uma vez na execução do programa, tornando-se visível de forma global para o restante do escopo do projeto.
- Foram criados três controllers, onde 1 estava relacionado a ItemCompravel (ItemCompravelController) e o outro estava relacionado às listas de compras

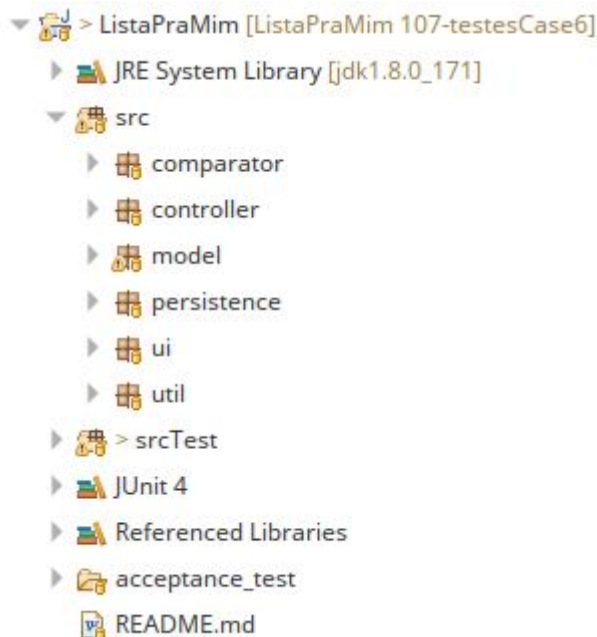
(ListaDeComprasController). Foi criado também o controller ListaPraMim, que estabelece a comunicação dos dois controllers mencionados com a interface (Facade), sendo assim um controller “mediador”.

## Estrutura do código

O código-fonte, na pasta src, foi separado em packages de responsabilidade, sendo o package controller que contém todos os Controllers do sistema, comparator contém todas as classes responsáveis por métodos de comparações que são usados em ordenação, o package model contém todas as classes que são representação de identidades do sistema. Package ui contém classes que realizam interação com usuário, e por fim, util contém classes que possuem métodos e atributos auxiliares.

O Folder source srcTest contém todos os testes de unidades do sistema, e cada um separado por packages correspondentes. A pasta acceptance\_test contém todos os testes de aceitação e o resources contém arquivos de persistência utilizados pelo sistema.

Imagem contendo a estrutura do código de acordo com o que foi explicado.



## Trabalho em grupo

No projeto, não houve separação de responsabilidades pré definida para cada pessoa, optamos por criar tasks e deixar que cada integrante do grupo pudesse escolher o que iria fazer. A cada caso de uso novo, era criado um projeto no github e cada integrante escolhia tasks que poderiam ser diferentes do seu trabalho no caso de uso interior. Isso possibilitou que todos aprendessem mais sobre tudo que foi ensinado na disciplina de P2 e LP2, como também, a troca

mútua de conhecimento no grupo. Dessa forma, todos os integrantes tiveram contato com todas as partes do projeto: herança, comparador, controller, facade, testes, validações.

A ferramenta utilizada para SCRUM foi o próprio github. Utilizando do Projects, Milestones e Issues para se criar os casos de uso, sprints e tasks, respectivamente. Algumas capturas de tela abaixo representam o que foi dito.

🚩 1 Open ✓ 2 Closed

Sort ▼

### Sprint 2

Closed 4 days ago ⌚ Last updated 4 days ago

100% complete 0 open 22 closed









[Edit](#) [Reopen](#) [Delete](#)

### Sprint 1

Closed 9 days ago ⌚ Last updated 9 days ago

100% complete 0 open 30 closed

[Edit](#) [Reopen](#) [Delete](#)

<input type="checkbox"/>	<div><div>🔴</div><b>VALIDAÇÃO URGENTE: Não deixar usuário alterar lista quando já finalizada</b></div> <div>#93 by thyannevlis was closed 4 days ago 🚩 Sprint 3</div> <div></div>	
<input type="checkbox"/>	<div><div>🔴</div><b>Cria estratégia de lista 3 automática</b></div> <div>#91 by thyannevlis was closed 3 hours ago 🚩 Sprint 3</div> <div></div>	
<input type="checkbox"/>	<div><div>🔴</div><b>Criar estratégia de lista 2 automática</b></div> <div>#90 by thyannevlis was closed 3 hours ago 🚩 Sprint 3</div> <div></div>	
<input type="checkbox"/>	<div><div>🔴</div><b>Criar estratégia 1 de lista automática</b></div> <div>#88 by thyannevlis was closed 4 days ago 🚩 Sprint 3</div> <div></div>	
<input type="checkbox"/>	<div><div>🔴</div><b>Ajustes do listaDeComprasController</b></div> <div>#79 by marianasn was closed 5 days ago</div> <div></div>	
<input type="checkbox"/>	<div><div>🔴</div><b>Realizar ajustes</b></div> <div>#77 by marianasn was closed 5 days ago</div> <div></div>	
<input type="checkbox"/>	<div><div>🔴</div><b>Pesquisando pelo código do item</b></div> <div>#76 by thyannevlis was closed 5 days ago 🚩 Sprint 2</div> <div></div>	
<input type="checkbox"/>	<div><div>🔴</div><b>Pesquisar listas usando data</b></div> <div>#75 by thyannevlis was closed 5 days ago 🚩 Sprint 2</div> <div></div>	

4 Open ✓ 5 Closed <span>Sort ▾</span>	
<b>Caso de uso 4: Pesquisar listas de compras</b> <span>Closed</span> Updated 2 hours ago	Listas de compras inseridas no Lista pra mim@ podem ser pesquisadas ...
<b>Caso de uso 3: CRUD de listas de compra</b> <span>Closed</span> Updated 2 hours ago	O app deve permitir a criação de listas de compra. ...
<b>Caso de uso 2: Pesquisa de itens</b> <span>Closed</span> Updated 13 days ago	outras funcionalidades de pesquisa de itens ...
<b>Caso de uso 1: CRUD dos itens compráveis</b> <span>Closed</span> Updated 14 days ago	Deve ser possível fazer CRUD (Create, read, update and delete) de itens compráveis. ... Todo item comprável deve manter as seguintes informações obrigatoriamente: identificador único numérico, nome, categoria e mapa de preços. O identificador numérico deve ser gerado automaticamente pelo sistema. Não deve ser permitido que itens iguais sejam adicionados no sistema. Dois itens compráveis são iguais se eles têm o mesmo nome e categoria. As categorias contempladas são: alimentos industrializados, alimentos não industrializados, limpeza e higiene pessoal.

A cada task pega por um integrante do grupo, era criada uma branch para task com o padrão: idDaTask-descricaoDaTask. Ao finalizar a task, o integrante fazia uma Pull Request onde era avaliado por pelo menos um integrante, que daria o merge da branch com o master. Esse modo de avaliação possibilitou que um integrante que estivesse fazendo os testes de unidade pudesse avaliar o controller criado pelo colega, por exemplo.

src/test/ItemControllerTest.java Hide outdated

```

19 + @Test(expected = IllegalArgumentException.class)
20 + public void cadastraItemNomeInvalido() {
21 +     controller.adicionaItemPorQtd("", "higiene pessoal", 4, "gr
22 +     controller.adicionaItemPorQtd(null, "alimento industrializa

```

siuannybsr

17 days ago

@marianasn, cria dois testes para esse cadastraltemNomeInvalido onde um testa vazio e outro testa null. Da forma como esse teste foi criado ele apenas está testando a linha 21 e retornando o erro nesse momento, ou seja, a linha 22 não é testada.

Reply...

✓

thayannevls approved these changes 3 hours ago

View changes

thayannevls commented 3 hours ago

+

😊

...










Perfect

thayannevls merged commit 0fcbabd into master 3 hours ago

Revert

# Resultados dos testes de cobertura

Obtivemos ótimos resultados nos testes de cobertura, tendo apenas pacotes não verificados que seriam o Facade que já é testado pelos testes de aceitação, e o util que contém apenas métodos auxiliares. No controller, ListaPraMim não possui teste pois é um controller intermediário, já sendo testado indiretamente pelos testes dos outros dois controllers.

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
▼ ListaPraMim	 80.4 %	5,820	1,423	7,243
▼ src	 73.8 %	2,697	957	3,654
▶ controller	 69.2 %	1,043	464	1,507
▶ ui	 0.0 %	0	252	252
▶ persistence	 0.0 %	0	121	121
▶ util	 91.3 %	748	71	819
▶ model	 93.6 %	719	49	768
▶ comparator	 100.0 %	187	0	187
▶ srcTest	 87.0 %	3,123	466	3,589

Todos os testes de aceitação passaram.

## STATISTICS:

Number of scripts: 12; with failures: 0

Number of tests: 425; with failures: 0

## Considerações finais

Não tivemos dificuldades ao desenvolver soluções para os casos de uso, já que o conhecimento ensinado na disciplina já era necessário para se realizar os trabalhos, sem apresentar grandes complexidades. Porém, muito tempo foi perdido por conta dos testes de aceitação por muitas vezes não seguirem a especificação, e ao longo da sprint, muitas mudanças tinham que ser feitas. Também às vezes se tornava difícil entender o que exatamente o caso de uso esperava do comportamento do programa. Essas dificuldades serviram de experiência para se imaginar como seria o processo de desenvolvimento com um cliente na vida real.

O método SCRUM foi de grande ajuda para o trabalho e organização em equipe, junto com as ferramentas do github para auxiliar.