André Alex Araujo Santos Camargo Pereira

Comparação da Testabilidade das Arquiteturas MVC e MVP na Camada de Apresentação em um aplicativo Android

Florianópolis – SC – Brasil

2017

André Alex Araujo Santos Camargo Pereira

Comparação da Testabilidade das Arquiteturas MVC e MVP na Camada de Apresentação em um aplicativo Android

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciências da Computação.

Universidade Federal de Santa Catarina

Departamento de Informática e Estatística

Curso de Ciências da Computação

Orientador: Raul Sidnei Wazlawick

Florianópolis – SC – Brasil

2017

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Pereira, André Alex Araujo Santos Camargo
Comparação da testabilidade das arquiteturas MVC e MVP na camada de apresentação em um aplicativo Android / André Alex Araujo Santos Camargo Pereira ; orientador, Raul Sidnei Wazlawick, 2018.
250 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Ciências da Computação, Florianópolis, 2018.

Inclui referências.

1. Ciências da Computação. 2. Testabilidade. 3. Model View Presenter. 4. Model View Controller. 5. Android. I. Wazlawick, Raul Sidnei. II. Universidade Federal de Santa Catarina. Graduação em Ciências da Computação. III. Título.

André Alex Araujo Santos Camargo Pereira

Comparação da Testabilidade das Arquiteturas MVC e MVP na Camada de Apresentação em um aplicativo Android

Monografia apresentada como requisito parcial para obtenção do grau de Bacharel em Ciências da Computação.

Trabalho aprovado. Florianópolis – SC – Brasil, 24 de novembro de 2012:

Raul Sidnei Wazlawick
Orientador

Patricia Vilain
Convidado 1

Rahony Goulart Ricardo
Convidado 2

Florianópolis – SC – Brasil
2017

Resumo

As vantagens de utilizar uma arquitetura que separe a lógica de apresentação e a manipulação de componentes de interface gráfica são desde uma melhor legibilidade e modularidade do código, o que facilita a manutenção, até uma melhora na testabilidade das classes.

Este trabalho comparou a testabilidade das arquiteturas MVC e MVP. Para essa comparação foi utilizada uma abordagem baseada em métricas de código fonte conforme discutidas em Bruntink (2003) (1).

Sendo concluído que se o objetivo é apenas se testar a lógica da apresentação, assumindo o risco de que algum componente gráfico seja apresentado de maneira incorreta, a versão MVP apresenta uma métrica de testabilidade melhor.

Palavras-chaves: Android. Aplicativo Mobile. Warehouse Management System. MVC. MVP. Testabilidade.

Lista de ilustrações

Figura 1 – FloatActionButton 31
Figura 2 – Cards 31
Figura 3 – Snackbar 31
Figura 4 – CollapsingToolbar 32

Figura 5 – Tela inicial 34
Figura 6 – Lista de Pedidos, a) Não processados, b) Processados 35
Figura 7 – Detalhes do Pedido, a) Itens restantes, b) Processamento finalizado, c) Já processado 36
Figura 8 – Detalhes do Pedido em Lote 37
Figura 9 – Tela de Processamento.a) Item reconhecido, b) Item já processado, c) Item inválido 37
Figura 10 – Lista de Pedidos, a) Não processados, b) Processados 38
Figura 11 – Tela de erro, mostrada quando houve problema para se comunicar com o serviço de armazenamento. 39
Figura 12 – Caixa de diálogo para confirmar ação e evitar que o usuário faça uma ação indesejada. 40

Figura 13 – Arquitetura Geral da Aplicação 42
Figura 14 – Exemplo Detalhes do Pedido 43
Figura 15 – Detalhes do Pedido Diagrama de Sequencia - Cenário de Sucesso 43

Lista de tabelas

Tabela 1 – Métrica por tela 64

Listagens

2.1 Exemplo Interface Presenter 30
2.2 Exemplo Interface View 30
4.1 Criação de um novo Filtro 44
4.2 Exemplo de OrderFilter 44
4.3 Adicionar método à Interface OrderFilter 45
4.4 Exemplo de método filter 45
4.5 Adicionando novo filtro à interface 46
4.6 Exemplo de conversão de tipo explicita 46
4.7 Activity que representa a Tela de Processamento 47
4.8 Interface OnItemProcessedListener 48
4.9 Exemplo onCreate 48
4.10 Método onItemProcessed 49
4.11 Versão reduzida do BarcodeProcessorContract 50
4.12 Implementando BarcodeProcessorContract.View 50
4.13 Implementando os métodos do BarcodeProcessorContract.View 51
4.14 View delegando para o Presenter 51
4.15 Implementando BarcodeProcessorContract.Presenter 52
4.16 Implementando os métodos do BarcodeProcessorContract.Presenter 52
5.1 Declaração de Mocks 55
5.2 Inicialização do teste do Presenter 56
5.3 Exemplo de teste do Presenter - caso de sucesso 56
5.4 Exemplo de teste do Presenter - caso de erro 57
5.5 Inicialização do teste da View 58
5.6 Exemplo de teste da View delegando para o Presenter 58
5.7 Exemplo de teste de View verificando informações na tela 59
5.8 Exemplo teste de caso de erro 59
5.9 Injeção de Repositório 61
5.10 Injeção com DataSource em Memória 61
5.11 Injeção com DataSource Falso 61
B.1 AndroidManifest 85
B.2 BaseActivity 86
B.3 BaseFragment 86
B.4 Empty State Adapter 86
B.5 Inject Prod 88
B.6 Inject Mock 88
B.7 ValueLabelView 88

B.8 App Build Gradle . . . 90
B.9 Project Build Gradle . . . 92
B.10 Barcode Processor Layout . . . 93
B.11 Main Layout . . . 94
B.12 Order Details Layout . . . 95
B.13 Order Info Layout . . . 96
B.14 Order Item Layout . . . 97
B.15 Orders List Layout . . . 97
B.16 Dashboard Layout . . . 98
B.17 Info Customer Layout . . . 99
B.18 Info Order Layout . . . 101
B.19 Info Shipment Layout . . . 102
B.20 Order Details Header Layout . . . 104
B.21 Processed Already Layout . . . 104
B.22 Processed Finish Layout . . . 105
B.23 Processed Items Layout . . . 105
B.24 Cards Empty State Layout . . . 106
B.25 Cards Order Items Layout . . . 106
B.26 Cards Orders List Layout . . . 108
B.27 View Value-Label . . . 111
B.28 CodesToProcess . . . 112
B.29 CustomerInfo . . . 113
B.30 NullOrderFilterList . . . 113
B.31 Order . . . 114
B.32 OrderFilterList . . . 115
B.33 OrderItem . . . 116
B.34 ShipmentInfo . . . 116
B.35 CommomOrdersRepository . . . 117
B.36 DataSourceCallback . . . 118
B.37 IdFilter . . . 119
B.38 OrderFilter . . . 119
B.39 OrdersDataSource . . . 120
B.40 OrdersFakeDataSource . . . 120
B.41 OrdersMemoryDataSource . . . 122
B.42 OrdersRepository . . . 125
B.43 OrderVisitor . . . 125
B.44 RepositoryCallback . . . 125
B.45 StatusFilter . . . 126
B.46 DateFormatterUtils . . . 126

B.47 MathUtils . . . 127
B.48 PermissionUtils . . . 127
B.49 StringUtils . . . 129
B.50 DialogUtils . . . 129
B.51 SnackBarUtils . . . 131
B.52 ImageLoader . . . 132
B.53 BarcodeProcessor . . . 132
B.54 BarcodeProcessorActivity . . . 133
B.55 CameraPreview . . . 138
B.56 DashboardFragment . . . 139
B.57 MainActivity . . . 141
B.58 OnItemProcessedListener . . . 142
B.59 OrderInfoActivity . . . 142
B.60 OrderItemActivity . . . 144
B.61 OrdersFactory . . . 146
B.62 OrdersListActivity . . . 147
B.63 OrdersListAdapter . . . 151
B.64 BarcodeProcessorActivityTest . . . 154
B.65 OrderDetailsActivityTest . . . 157
B.66 OrdersListActivityTest . . . 168
B.67 BaseView . . . 175
B.68 BasePresenter . . . 175
B.69 BarcodeProcessor . . . 176
B.70 BarcodeProcessorActivity . . . 177
B.71 BarcodeProcessorContract . . . 181
B.72 BarcodeProcessorPresenter . . . 182
B.73 CameraPreview . . . 184
B.74 DashboardContract . . . 184
B.75 DashboardFragment . . . 185
B.76 DashboardPresenter . . . 187
B.77 MainActivity . . . 189
B.78 OnItemProcessedListener . . . 189
B.79 OrderDetailsAdapter . . . 189
B.80 OrderDetailsAdapter . . . 197
B.81 OrderDetailsContract . . . 199
B.82 OrderDetailsPresenter . . . 201
B.83 OrderInfoActivity . . . 204
B.84 OrderInfoContract . . . 207
B.85 OrderInfoPresenter . . . 207

B.86 OrderItemActivity 208
B.87 OrderItemContract 210
B.88 OrderItemPresenter 211
B.89 OrdersListActivity 211
B.90 OrdersListAdapter 215
B.91 OrdersListContract 217
B.92 OrdersListPresenter 218
B.93 BarcodeProcessorActivityTest 220
B.94 BarcodeProcessorPresenterTest 222
B.95 OrderDetailsActivityTest 225
B.96 OrderDetailsPresenterTest 234
B.97 OrdersListActivityTest 242
B.98 OrdersListPresenterTest 247

Lista de abreviaturas e siglas

MVC	Model-View-Controller
MVP	Model-View-Presenter
RFC	Response For Class
LOC	Lines Of Code
NOTC	Number of Test Cases
tLOC	Lines Of Code for Test Classes

Sumário

Listagens **11**

1 Introdução **21**

Introdução **21**

1.1 Justificativa 21

1.2 Objetivos 22

1.2.1 Objetivo Geral 22

1.2.2 Objetivos Específicos 22

1.3 Organização do Trabalho 23

2 Fundamentação Teórica **25**

2.1 Testabilidade 25

2.2 Android 26

2.2.1 Layout 26

2.2.2 Activities 27

2.3 Arquiteturas de Camada de Apresentação 27

2.3.1 Model View Controller - MVC 27

2.3.2 Model View Presenter - MVP 27

2.3.2.1 Passive View 28

2.3.2.2 Supervising Controller 28

2.3.3 Model View View Model - MVVM 29

2.4 Arquiteturas de Camada de Apresentação no Desenvolvimento Android . . 29

2.4.1 Model View Controller 29

2.4.2 Model View Presenter 29

2.5 Material Design 30

2.5.1 FloatActionButton 30

2.5.2 Cards 31

2.5.3 Snackbar 31

2.5.4 CollapsingToolbar 32

3 Caso de Estudo **33**

3.1 Visão Geral 33

3.2 Motivação para a aplicação 33

3.3 A Aplicação 34

3.3.1 Tela Inicial 34

3.3.2 Tela de Lista de Pedidos . . . 35
3.3.3 Tela de Detalhes do Pedido . . . 36
3.3.3.1 Processamento Individual . . . 36
3.3.3.2 Processamento em Lote . . . 36
3.3.4 Tela de Processamento . . . 37
3.3.5 Heurísticas de Nielsen . . . 38
3.3.5.1 Visibilidade do estado do sistema . . . 38
3.3.5.2 Ajudar o usuário a reconhecer, diagnosticar e se recuperar de erros . . . 38
3.3.5.3 Prevenção de erros . . . 39

4 Desenvolvimento . . . 41
4.1 Arquitetura da Aplicação . . . 41
4.1.1 Exemplo: Detalhe do Pedido . . . 42
4.2 Utilização de Padrões de Projetos . . . 43
4.2.1 Filtros e o Padrão Visitor . . . 44
4.2.1.1 Passo 1: Criar o filtro . . . 44
4.2.1.2 Passo 2: Adicionar o filtro na interface OrderVisitor . . . 45
4.2.1.3 Passo 3: Implementar o novo filtro em todos os OrderVisitor 45
4.2.1.4 Passo 4: Adicionar o filtro à interface . . . 46
4.2.1.5 Demais explicações . . . 46
4.3 A Camada de Apresentação . . . 47
4.3.1 Criação do Layout . . . 47
4.3.2 Criação da Activity com a arquitetura MVC . . . 47
4.3.3 Refatoração da Activity para a arquitetura MVP . . . 49
4.3.3.1 Extração do Contrato . . . 50
4.3.3.2 Criação da View . . . 50
4.3.3.3 Criação do Presenter . . . 52

5 Testes . . . 55
5.1 Versão MVP . . . 55
5.1.1 Testando o Presenter . . . 55
5.1.2 Testando a View . . . 58
5.2 Versão MVC . . . 59
5.2.1 Testando a Activity . . . 59
5.2.2 Criando um DataSource Falso . . . 60

6 Aplicação da Métrica de Testabilidade . . . 63
6.1 Análise das métricas de Teste . . . 63
6.2 Análise das métricas de Código Fonte . . . 64

6.3 Considerações 65

7 Conclusão 67

7.1 Trabalhos Futuros 68

Referências 69

Anexos 73

ANEXO A Artigo 75

ANEXO B Código Fonte 85

B.1 Código Comum entre as versões MVC e MVP 85

B.1.1 Layouts 93

B.1.2 Entity 112

B.1.3 Repository 117

B.1.4 Utils 126

B.2 Código Específico versão MVC 132

B.2.1 Testes 154

B.3 Código Específico versão MVP 175

B.3.1 Testes 220

1 Introdução

Uma tela no desenvolvimento para a plataforma Android tem sua estrutura definida em um arquivo XML e os componentes definidos são referenciados em uma classe que herda de Activity. Esta classe, ao se desenvolver utilizando a arquitetura Model View Controller, conversa com a camada de domínio (Model) e controla os elementos de interface gráfica para refletirem o estado dos objetos do modelo. Assim sendo uma Activity, na arquitetura MVC, possui tanto responsabilidades de View - referências para os componentes de tela - quanto responsabilidades de Controller - refletir ações com a interface gráfica no modelo e comandá-la para mostrar o estado correto dos objetos de domínio na tela.

Desta forma uma Activity acaba ficando com muitas responsabilidades:

- Receber ações do usuário.
- Tratar os dados de entrada.
- Se comunicar com a camada Model para atualizar ou recuperar algum dado.
- Tratar o dado recebido para ser apresentado na tela.
- Atualizar o elemento da View com o novo dado.

Todas essas responsabilidades já tornam uma Activity muito difícil de ser testada. Mas, além disso, o fato de uma Activity fazer parte do SDK do Android, os testes ou devem ser realizados em um aparelho real, um emulador ou utilizar algum framework como o Roboletic que possibilita o teste de componentes do SDK do Android serem testados na JVM. Complicando ainda mais os testes dessas classes, soma-se a isso tudo o fato de as Activities terem um ciclo de vida gerido pelo Android.

Alguns padrões de camada de apresentação podem ser utilizados afim de tirar algumas responsabilidades da Activity, sendo um deles o Model View Presenter.

1.1 Justificativa

Ao utilizar a arquitetura Model View Presenter a View se torna passiva, apenas notificando o Presenter sobre as interações do usuário passando os dados necessários e recebendo comandos para atualizar a tela. A lógica do que fazer a partir de uma interação é delegada ao Presenter.

Com essa separação de responsabilidades os seguintes benefícios podem ser notados:

- **Legibilidade do Código:** ao olhar apenas a interface exposta pela View é possível saber quais ações ela é capaz de executar.
- **Modularidade:** para trocar uma View por outra, basta apenas que a nova implemente a mesma interface que a antiga.
- **Testabilidade:** com uma implementação eficiente na plataforma Android, pode-se fazer com que o Presenter seja totalmente independente do Framework. Facilitando, assim, o teste dos componentes.

Tendo em vista que uma melhor Testabilidade é uma vantagem da arquitetura MVP esse trabalho pretende verificar essa afirmação comparando métricas de código fonte de um mesmo aplicativo implementados utilizando as duas arquiteturas.

Apesar de uma das vantagens, na hora de realizar os testes, de se utilizar o padrão MVP seja o fato de os testes poderem rodar diretamente na JVM, sem necessidade de um aparelho real ou emulador, o escopo deste trabalho se limita à comparação entre as arquiteturas MVC e MVP em métricas de código fonte e métricas baseadas no código de teste.

1.2 Objetivos

1.2.1 Objetivo Geral

Comparar a testabilidade de um aplicativo Android utilizando Model View Controller e Model View Presenter como arquiteturas da camada de apresentação.

1.2.2 Objetivos Específicos

- Sumarizar as principais arquiteturas de camada de apresentação;
- Desenvolver um aplicativo Android que auxilie na montagem de pedidos recebidos por lojas virtuais, para servir de caso de estudo para aplicação e comparação das métricas de testabilidade;
- Desenvolver teste de unidade para o aplicativo implementado, para servir de caso de estudo para aplicação e comparação das métricas de testabilidade;
- Comparar a testabilidade das arquiteturas MVC e MVP, do caso de estudo desenvolvido, utilizando métricas baseadas em código fonte;

- Comparar a testabilidade das arquiteturas MVC e MVP, do caso de estudo desenvolvido, utilizando métricas baseadas no código de teste.

1.3 Organização do Trabalho

O trabalho inicia com um capítulo de Fundamentação Teórica. Primeiramente é apresentado o conceito de Testabilidade e as métricas utilizadas para medi-la.

Logo após é explicado quais são os principais componentes da camada de apresentação utilizados no Framework Android. E em seguida é mostrado como cada uma dessas peças são organizadas nas arquiteturas implementadas.

O capítulo 3 fala sobre a aplicação desenvolvida, sua motivação, detalhes de funcionamento e sua usabilidade.

No capítulo 4 são mostrados detalhes de implementação do aplicativo, sua arquitetura e camada de apresentação.

Explicações sobre a criação dos testes para as versões Model-View-Controller e Model-View-Presenter são comentadas no capítulo 5.

Ao final do trabalho, capítulo 6, a testabilidade das versões são comparadas utilizando a abordagem de Bruntink (2003) (1).

Após o trabalho é finalmente concluído, no capítulo 7, avaliando os objetivos propostos e alcançados e sugerindo trabalhos futuros.

2 Fundamentação Teórica

2.1 Testabilidade

A ISO/IEC 25010 define testabildade como:

> Grau de eficácia e eficiência com o qual critérios de teste podem ser estabelecidos para um sistema, produto ou componente e testes podem ser realizados para determinar se esses critérios foram atendidos.

Bruntink (1) investiga a testabilidade na perspectiva de Testes de Unidade. Ele estuda um conjunto de métricas que podem ser usadas para avaliar a testabilidade de classes Java.

Em seu trabalho Bruntink (1) investiga as métricas a seguir:

- DIT - Depth Of Inheritance Tree
- FOUT - Fan Out
- LCOM - Lack Of Cohesion Of Methods
- LOCC - Lines Of Code Per Class
- NOC - Number Of Children
- NOM - Number Of Methods
- RFC - Response For Class
- WMC - Weighted Methods Per Class

Primeiramente ele define o esforço para se testar uma classe como sendo o tamanho do conjunto de testes. E decide utilizar as seguintes métricas:

- **tLOC - Lines Of Codes:** O número de linhas de código necessários para testar a classe.
- **NOTC - Number of Test Cases:** O número de casos de testes necessários para testar a classe.

Então em seu trabalho ele tenta responder a seguinte questão:

> Os valores das métricas baseadas em código fonte para uma classe tem correlação com o número de linhas de código e o número de casos de testes do conjunto de testes correspondente?

A fim de responder essa questão ele faz dois estudos de caso. Com o software DocGen, um gerador de documentação, e outro com o Apache Ant uma ferramenta de build.

Ao final do seu estudo foram encontradas algumas relações moderadamente fortes para cada caso de estudo, são elas:

- **DocGen:** FOUT,LOCC e RFC
- **Apache Ant:** FOUT, LCOM, LOCC, NOF, NOM, RFC e WMC

Neste trabalho foram utilizadas como medida de testabilidade de uma classe apenas as métricas que demonstraram uma correlação moderadamente forte nos dois estudos de caso. São elas:

- **LOCC - Lines Of Code Rer Class:** O número de linhas de código de uma classe, desconsiderando linhas em branco e comnetários
- **RFC - Response For Class:** O número de métodos potencialmente invocados por uma classe.

2.2 Android

Conforme explica a documentação do Android(2), um componente é um bloco de construção de um aplicativo. Sendo que cada componente é um ponto diferente pelo qual o sistema pode entrar no aplicativo.

Dentre estes componentes está a Activity que representa uma tela com uma interface com o usuário. Um Layout define a estrutura visual de uma Activity. As seções a seguir comentam mais sobre Layout e Activity.

2.2.1 Layout

Arquivos de layout definem a estrutura da interface. Esses arquivos são definidos utilizando a linguagem XML.

Cada arquivo deve conter exatamente um elemento raiz. Esse elemento pode ser uma View, como um TextView ou ImageView, ou então um ViewGroup, como LinearLayout ou FrameLayout, que são views que podem conter filhos que podem ser outros ViewGroup ou outras View.

Cada tipo ViewGroup define a forma como os filhos serão renderizados na tela. E cada View define um tipo de elemento que pode mostrar um texto, uma imagem, um campo para entrada de texto, entre outros.

Combinando diversos ViewGroup e View o desenvolvedor pode criar as interfaces da aplicação.

2.2.2 Activities

As Activities são classes escritas na linguagem Java que herdam de Activity. São nessas classes que são carregados os layouts.

Depois de carregar um layout a Activity pode conseguir acesso aos seus componentes. Podendo assim alterar algum texto de algum TextView ou recuperar o texto digitado em um EditView.

Ao herdar de Activity o Framework trata a classe como uma tela da aplicação que pode ser aberta como a tela inicial ou a partir de uma ação executada em outra tela. Assim como passa a respeitar um ciclo de vida específico que pode ser visto em Android Activity(3).

2.3 Arquiteturas de Camada de Apresentação

Essa seção faz um resumo das principais arquiteturas de camada de apresentação.

2.3.1 Model View Controller - MVC

Segundo Martin Fowler (4) as principais características do padrão Model View Controller são:

- Separar a apresentação (View e Controller) do domínio (Model).
- O controller é responsável por reagir aos eventos do usuário.
- A View é responsável por renderizar o estado do Model na tela.

Porém um ponto fraco deste padrão é que a View é responsável pela lógica de apresentação de um objeto do modelo. Por exemplo, a View pode conter um texto que deve ser colorido com uma cor ou outra, dependendo de algum dado do Model.

2.3.2 Model View Presenter - MVP

De acordo com Fowler (5) a arquitetura Model View Presenter deve ser dividida entre Supervising Controller (6) e Passive View (7). As particularidades de cada uma são

comentadas nas próximas seções. No restante deste trabalho a arquitetura Model View Presenter se refere à variante Passive View proposta por ele.

Ainda segundo Fowler os frameworks de interface gráfica não foram construídos com a preocupação de serem testáveis. O padrão Model-View-Presenter, em suas variantes, tem como objetivo facilitar a camada de apresentação.

O padrão MVP possibilita a utilização de um mock para a View nos testes do Presenter, assim os testes podem ser realizados sem a interação com o framework de interface gráfica.

2.3.2.1 Passive View

Uma Passive View (7) lida com o problema de testabilidade das camadas de apresentação reduzindo a lógica dos componentes de interface ao mínimo. Logo as responsabilidade da View são:

- Receber ações do usuário.
- Passar a ação para o Presenter juntamente com o dado de entrada.
- Receber ações do Presenter com os dados já formatados e apenas apresentá-los na tela.

Já as responsabilidades do Presenter são:

- Receber os eventos da View e seus dados.
- Se comunicar com a camada Model para atualizar ou recuperar algum dado.
- Chamar ações na View com os dados já formatado.

2.3.2.2 Supervising Controller

O padrão Supervising Controller é muito parecido com o Passive View, a diferença se dá pelo fato de que atualizações simples no Model, que não necessitem de um tratamento adicional pelo Presenter, podem ser diretamente sincronizados com a View através do padrão Observer, por exemplo.

De acordo com Martin Fowler (4) a utilização do padrão Observer tem a desvantagem de dificultar a compreensão do que está acontecendo apenas lendo o código.

A escolha entre essas duas formas da arquitetura MVP, Passive View ou Supervising Controller, deve balancear o fato de que na primeira variante mais código deve ser escrito porém na segunda a lógica colocada na View - ainda que simples - não será testável através do Presenter.

2.3.3 Model View View Model - MVVM

O padrão MVVM é chamado de Presentation Model por Fowler (8) e é implementado através de uma classe que representa os dados e o comportamento da View mas não é encarregada de redenrizar os dados na tela. Tal responsabilidade fica à cargo da View que observa, através do padrão Observer, este Presentation Model.

Assim como o padrão MVP o MVVM retira a lógica da View, possibilitando o teste apenas do Presentation Model sem precisar interagir com o framework de interface gráfica.

2.4 Arquiteturas de Camada de Apresentação no Desenvolvimento Android

Na seção anterior as principais arquiteturas da camada de apresentação foram apresentadas. Nesta seção as arquiteturas MVC e MVP serão comentadas no contexto do desenvolvimento Android.

2.4.1 Model View Controller

No padrão MVC aplicado ao desenvolvimento Android o arquivo de layout faz parte da View e a Activity do Controller. Eles são ligados da seguinte forma:

- O arquivo de layout define a interface da View
- Interações com o usuário como o clique de um botão são tratadas na Activity.
- Como a Activity faz o papel de Controller, ela mesmo é responsável por saber que ações devem ser tomadas a partir das interações. Como por exemplo ela é responsável por recuperar os campos preenchidos em um formulário, atualizar esses dados no modelo e tomar alguma atitude depois de o modelo validar os dados.

Mais detalhes sobre a implementação da arquitetura Model View Controller no Android pode ser encontrada em Madhvani (2016) (9).

2.4.2 Model View Presenter

No padrão MVP aplicado ao desenvolvimento Android o arquivo de layout e a Activity fazem parte da View. Eles são ligados da seguinte forma:

- O arquivo de layout define a interface da View

- A Activity é responsável por receber interações do usuário, recuperar os dados e enviá-los para o Presenter.

- O Presenter atualiza o modelo e é responsável por atualizar a View depois de o modelo validar os dados.

Os métodos do Presenter costumam ser nomeados de forma passiva, conforme mostra a Listagem 2.1:

Listagem 2.1 – Exemplo Interface Presenter

```
void onFormConfirmation(String nome, String email);
void onUploadCancel();
```

Os métodos da View costumam ser nomeados de forma imperativa, como na Listagem 2.2:

Listagem 2.2 – Exemplo Interface View

```
void showError(String error);
void removeLoadingView();
```

Mais detalhes sobre a implementação da arquitetura Model View Presenter no Android pode ser encontrada em Madhvani (2016) (9).

2.5 Material Design

Material Design (10) é o Guia de Estilos indicado pelo Google para os aplicativos Android (11). A seguir serão apresentados alguns componentes desse guia que foram utilizados na implementação desse trabalho.

2.5.1 FloatActionButton

De acordo com (12) um FloatActionButton mostra a ação primária na aplicação. A Figura 1 é de um FloatActionButton.

Figura 1 – FloatActionButton

2.5.2 Cards

Como pode ser visto em (13) os Cards podem conter imagem, texto e link sobre um assunto. A Figura 2 mostra um Card.

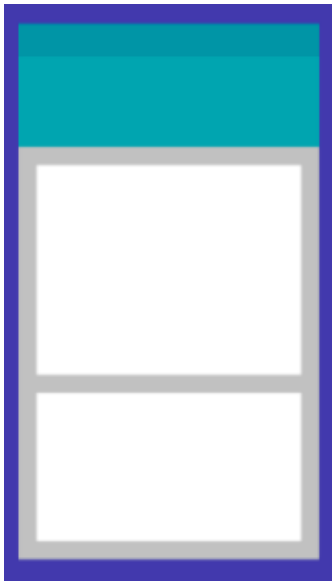


Figura 2 – Cards

2.5.3 Snackbar

Conforme explicado em (14) um Snackbar mostra uma mensagem sobre alguma operação através de uma mensagem na parte de baixo da tela. A Figura 3 é um exemplo de um Snackbar.

Figura 3 – Snackbar

2.5.4 CollapsingToolbar

De acordo com (15) uma CollapsingToolbar oferece características visuais e interações para a barra de ferramentas. A Figura 4 mostra um CollapsingToolbar.

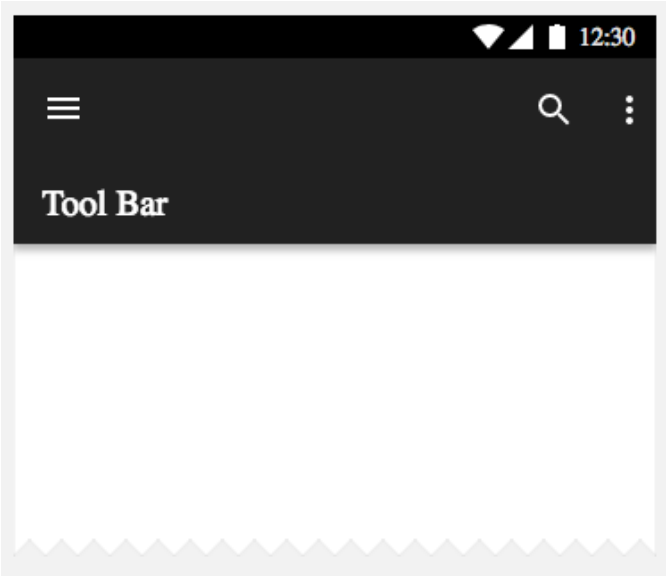


Figura 4 – CollapsingToolbar

3 Caso de Estudo

3.1 Visão Geral

segundo Medeiros (1999) (16) Para ter uma loja, os produtos que ela vende devem ser armazenados em algum lugar. Todo tipo de armazém deve ser capaz de receber, armazenar, coletar e preparar para entrega seus produtos.

Ainda segundo Medeiros, a atividade de coleta dos produtos em suas quantidades corretas é considerada uma das mais críticas de um armazém.

Eduardo de Souza Canal (17), escreveu:

> "A implantação de um Warehouse Management System (WMS) utilizando o módulo automatizado (com leitor de código de barras) atua diretamente nos erros citados acima. Fazendo o uso do coletor de dados no picking, o sistema valida se mesmo item que está sendo lido é o solicitado, no momento da leitura do código de barras do produto, evitando assim que aconteça a separação de objetos incorretos."

E continua:

> "Por padrão, o sistema obriga que a leitura dos códigos de barras dos produtos seja realizada para a confirmação da separação. No processo seguinte, de conferência de expedição, é possível detectar a falta do produto e verificar de qual endereço foi confirmada sistemicamente a separação, anulando a possibilidade de esquecimento de itens."

3.2 Motivação para a aplicação

Sistemas de Gerenciamento de Armazéns são muito complexos, a fim de gerenciar um grande armazém, e com um preço elevado - vide (18) com preço inicial de $4395 (por licença sem expiração para 1 usuário).

Um pequeno vendedor que necessita de ajuda na hora de preparar seus produtos para entrega, não precisa gerir um grande armazém e acaba tendo que pagar por funcionalidade que não irá usar ou conferindo seus pedidos de maneiras menos confiáveis.

A não utilização de nenhum método de controle na hora de montagem de pedidos torna essa tarefa ineficiente e propensa a erros.

3.3 A Aplicação

A aplicação foi feita seguindo o Guia de Estilos Material Design (10), recomendado pelo Google. As Heurísticas de Nielsen (19) também foram levadas em consideração para o desenvolvimento da interface.

Nas seções a seguir será apresentada uma visão geral das telas do aplicativo e como elas ajudam a gerenciar um pequeno armazém. Assim como serão destacados componentes do Material Design utilizados e algumas Heurísticas de Nielsen.

3.3.1 Tela Inicial

Na tela inicial do aplicativo (Figura 5), encontram-se diversos tipos de filtros para facilitar o processamento dos pedidos. São eles:

- **Não Processados:** Exibe uma lista com os itens que ainda não foram processados, facilitando o processamento em lotes (como será visto na tela de detalhes).
- **Processados:** Exibe uma lista com os itens que já foram processados, facilitando a revisão dos pedidos.
- **Todos:** Exibe uma lista com todos os pedidos, incluindo os pedidos já processados e os que ainda não foram. Esse filtro facilita ter uma visão geral dos pedidos.
- **Por id:** Caso seja necessário encontrar uma lista de pedidos com IDs específicos, esse filtro deve ser utilizado.

Figura 5 – Tela inicial

Os filtros são muito importantes nesse tipo de aplicação, pois a pessoa que está montando o pedido pode querer, por exemplo, montar primeiro os pedidos com determinados IDs pois são pedidos especiais. Ou pode ser interessante montar primeiro pedidos que utilizam determinado tipo de transportadora, pois necessitam de uma embalagem especial.

Apesar de a aplicação atualmente contar apenas com os 4 filtros mencionados na lista acima, a criação de novas formas de filtragem de resultado foi desenvolvida de modo que seja fácil adicioná-las na interfaces, a partir do momento que elas foram implementadas no sistema de armazenamento. A implementação dos filtros será discutida mais adiante nesse trabalho.

3.3.2 Tela de Lista de Pedidos

Após a escolha de um filtro na Tela Inicial é exibida a tela com os pedidos resultantes de sua aplicação.

Figura 6 – Lista de Pedidos, a) Não processados, b) Processados

Na Figura 6 a) existe um Float Action Button(12) no canto inferior direito para o processamento em lote.

Ao selecionar um Card(13), em ambas as listas, a tela de Detalhes do Pedido correspondente ao item selecionado é aberta. Sendo que no caso de o produto já ter sido processado (b) os detalhes são mostrados, porém não há a possibilidade de abrir a câmera para reprocessá-lo.

3.3.3 Tela de Detalhes do Pedido

3.3.3.1 Processamento Individual

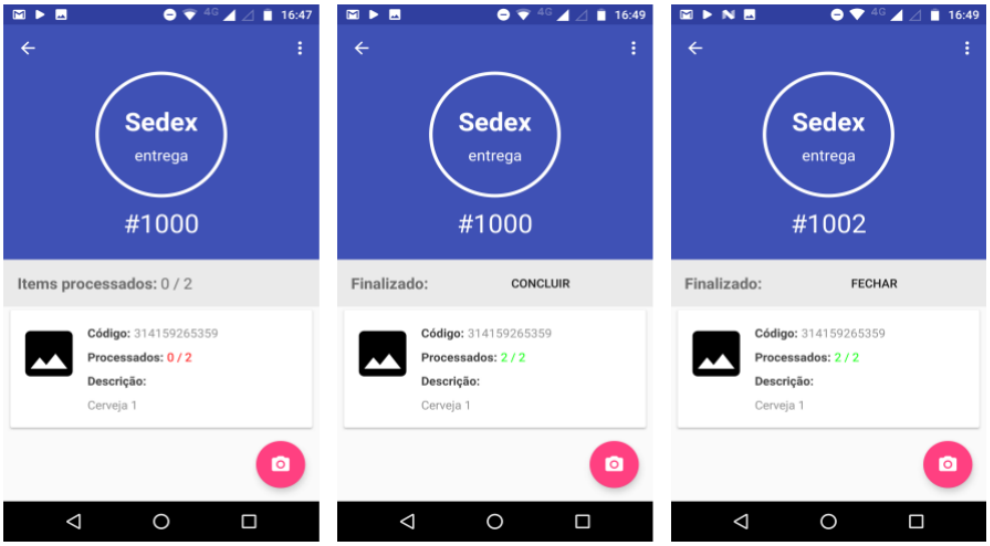


Figura 7 – Detalhes do Pedido, a) Itens restantes, b) Processamento finalizado, c) Já processado

Na Figura 7 a) existe um Float Action Button(12) no canto inferior direito para iniciar o processamento do pedido. Na Collapsing Toolbar(15) é exibido o ID do pedido e o tipo de entrega.

Abaixo é exibida a lista com os itens que fazem parte do pedido. Exibindo informações como foto do produto, código e uma descrição. Assim como a quantidade de itens daquele produto fazem parte do pedido, e quantos já foram processados.

Entre a Collapsing Toolbar (15) e a lista há uma tarja que pode ser das formas mostradas nas Figura 7 a),b) e c).

- **Figura 7 a):** Pedido ainda não foi processado e ainda possuí itens pedentes de processamento
- **Figura 7 b):** Pedido ainda não foi processado, porém todos itens já foram processados. E o pedido pode ser finalizado.
- **Figura 7 c):** O pedido já foi processado.

3.3.3.2 Processamento em Lote

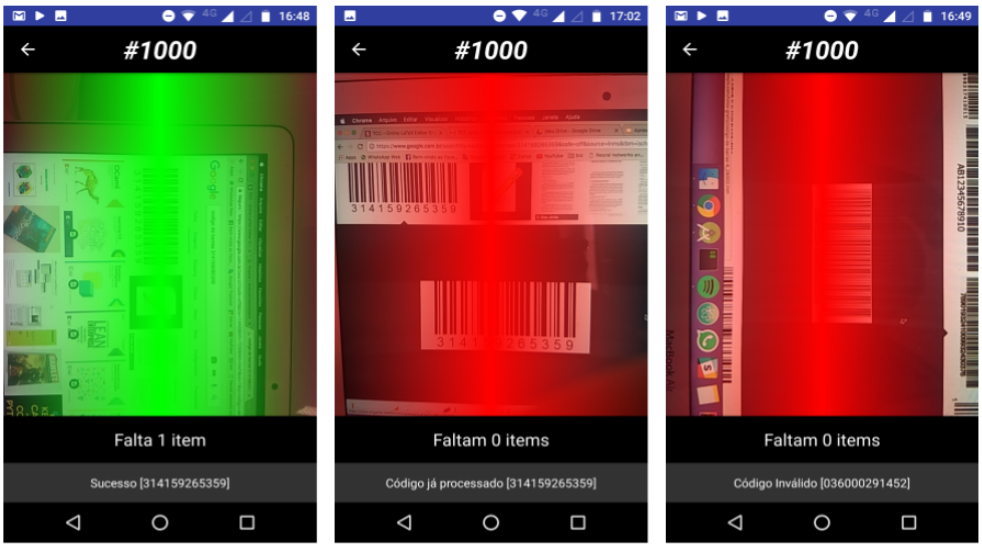
Ao ser iniciada no modo de Processamento em Lote a tela de Detalhes do Pedido (Figura 8) exibe na Collapsing Toolbar (15) a contagem do total de pedidos a serem

processados e qual é o pedido atual. E exibe, também, um botão de PULAR para desconsiderar o pedido atual e passar para o próximo. O restante das funções são idênticas ao Processamento Individual.

Figura 8 – Detalhes do Pedido em Lote

3.3.4 Tela de Processamento

Iniciada a partir da tela de Detalhes do Pedido, é responsável por reconhecer os códigos de barras dos produtos correspondente ao pedido que à iniciou.



a) b) c)

Figura 9 – Tela de Processamento.a) Item reconhecido, b) Item já processado, c) Item inválido

Basta focar a câmera para o código de barras de um produto, caso o produto faça parte do pedido a tela tomará a forma da Figura 9 a). Caso o mesmo produto seja processado um número maior de vezes do que a quantidade pedida o aviso da 9 b) será mostrado. Finalmente, se o código lido não fizer parte do pedido, será exibida a mensagem da 9 c). Os avisos utilizam o componente Snackbar(14).

O usuário poderá ir e voltar da tela de Detalhes do Pedido sem perda de informações. Quando todos os itens forem processados a tarja conforme Figura 7 b) será exibida.

3.3.5 Heurísticas de Nielsen

Seguem alguns exemplos de conformidade da aplicação desenvolvida com as Heurísticas de Nielsen.

3.3.5.1 Visibilidade do estado do sistema

Segundo esta heurística o sistema deve manter o usuário informado sobre o que está acontecendo no sistema.

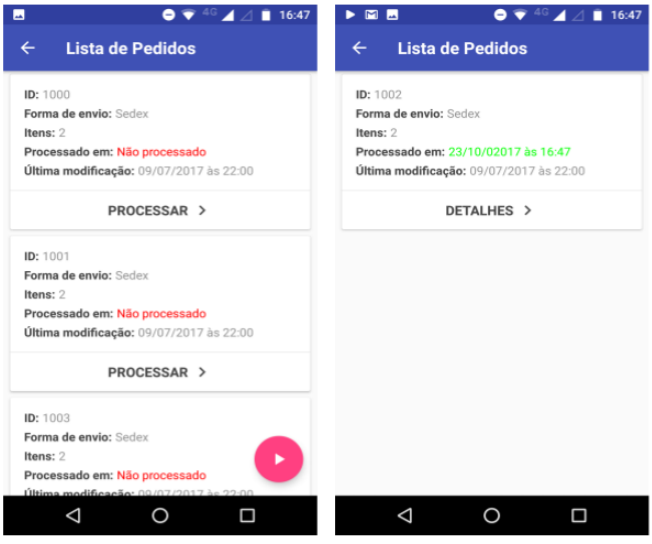


Figura 10 – Lista de Pedidos, a) Não processados, b) Processados

Pode-se notar que os itens processados aparecem com uma tarja verde e com a ação 'DETALHES', enquanto os itens não processados aparecem como uma tarja verde e a ação 'PROCESSAR'.

3.3.5.2 Ajudar o usuário a reconhecer, diagnosticar e se recuperar de erros

Esta heurística dita que mensagens de erro devem ser expressas utilizando linguagem facilmente entendida por humanos, isto é, não utilizar códigos. A Figura 11 mostra

um exemplo da sua aplicação no aplicativo desenvolvido, onde no momento em que uma lista não pode ser carregada é exibido uma mensagem que o motivo foi por falha de comunicação, o que indica um problema com a internet do usuário.

Figura 11 – Tela de erro, mostrada quando houve problema para se comunicar com o serviço de armazenamento.

3.3.5.3 Prevenção de erros

De acordo com a heurística de prevenção de erros, melhor do que mostrar uma boa mensagem de erro é evitar que o erro ocorra. A Figura 12 mostra o emprego da heurística no momento em que o usuário decide fechar o processamento de um pedido, ao ser exibida uma caixa de diálogo pedindo a confirmação de uma ação destrutiva - todo progresso pode ser perdido - visto que o botão de fechar pode ter sido selecionado por engano.

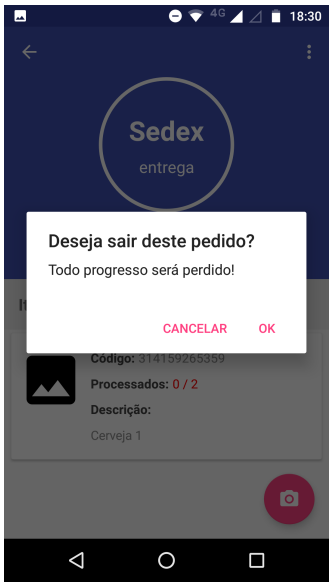


Figura 12 – Caixa de diálogo para confirmar ação e evitar que o usuário faça uma ação indesejada.

4 Desenvolvimento

O aplicativo foi desenvolvido utilzando a linguagem Java e o Framework Android. Ele possuí as seguintes camadas:

- **Apresentação (presentation):** Responsável por apresentar as informações para o usuário e tratar suas interações.
- **Repositório (repository):** 'Homem do meio' entre a apresentação e a fonte de dados. Pode, por exemplo, ser responsável por cachear dados e verificar se os dados estão na cache antes de emitir uma chamada para uma fonte de dados.
- **Fonte de dado (data source):** É quem de fato recupera o dado, pode ser através de uma chamada de API, banco de dados ou até em memória.
- **Entidade (entity):** É o objeto retornado pela fonte de dados para o repositório e então para a camada de apresentação. Representa uma entidade do domínio, como um Pedido.

O foco do desenvolvimento foi a camada de apresentação (Figura 13). Para tal a arquitetura da aplicação deveria ser robusta o suficiente para que uma mudança na forma de armazenamento e recuperação dos dados, pelas camadas inferiores, não afetassem a camada de apresentação.

4.1 Arquitetura da Aplicação

A Figura 13 representa a Arquitetura Geral da aplicação. Nela a letra I maiúscula no início do nome do componente indica que ele é uma interface.

Portanto a aplicação segue o Princípio da Inversão de Dependências (20), onde uma dependência deve ser de uma abstração, não de uma implementação concreta. Como é difícil garantir que todas as dependências sejam de abstrações, esse princípio, como pode ser visto através da figura, é seguido principalmente quando são cruzadas camadas da aplicação. Por exemplo na camada de apresentação depende-se da interface de um repositório. Porém caso altere-se a implementação do repositório para utilizar um cache e evitar chamadas na rede ou banco de dados, a camada de apresentação não deve se alterar.

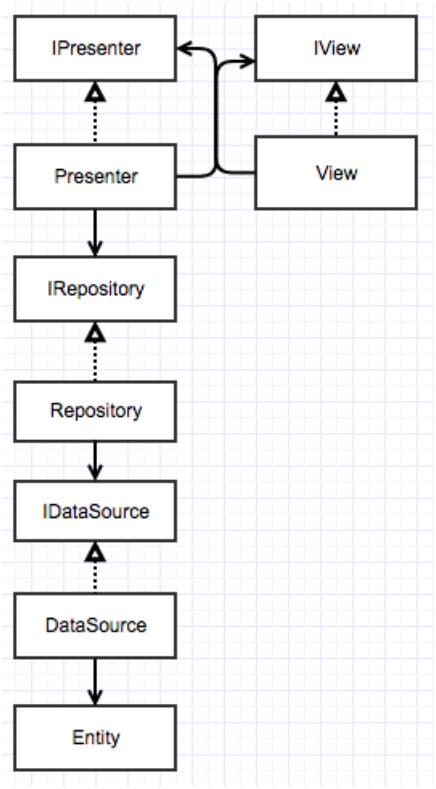


Figura 13 – Arquitetura Geral da Aplicação

4.1.1 Exemplo: Detalhe do Pedido

Conforme comentado na abertura desse capítulo, o desenvolvimento desse trabalho foi focado na camada de apresentação. Portanto o armazenamento de dados é feito em memória, como pode ser visto na Figura 14 em OrdersMemoryDataSource. Caso seja, futuramente, implementado um serviço de consumo de API ou um banco de dados local, apenas deve-se injetar em CommonOrdersRepository o DataSource desejado, as camadas de repositório e apresentação não se alteram. Como disse Robert Martin em (21):

> "Databases and frameworks are details! You don't have to decide upon them up front."

Na Figura 15 é apresentado um diagrama de sequência para um cenário de sucesso ao obter um pedido específico. É importante ressaltar a utilização de callbacks entre as camadas. A camada de fonte de dados deve fazer seu trabalho em background, deixando a thread principal livre para poder interagir com o usuário. Inclusive o Framework Android lançará uma exceção caso seja feito um acesso à rede na thread principal (NetworkOnMainThreadException (22))

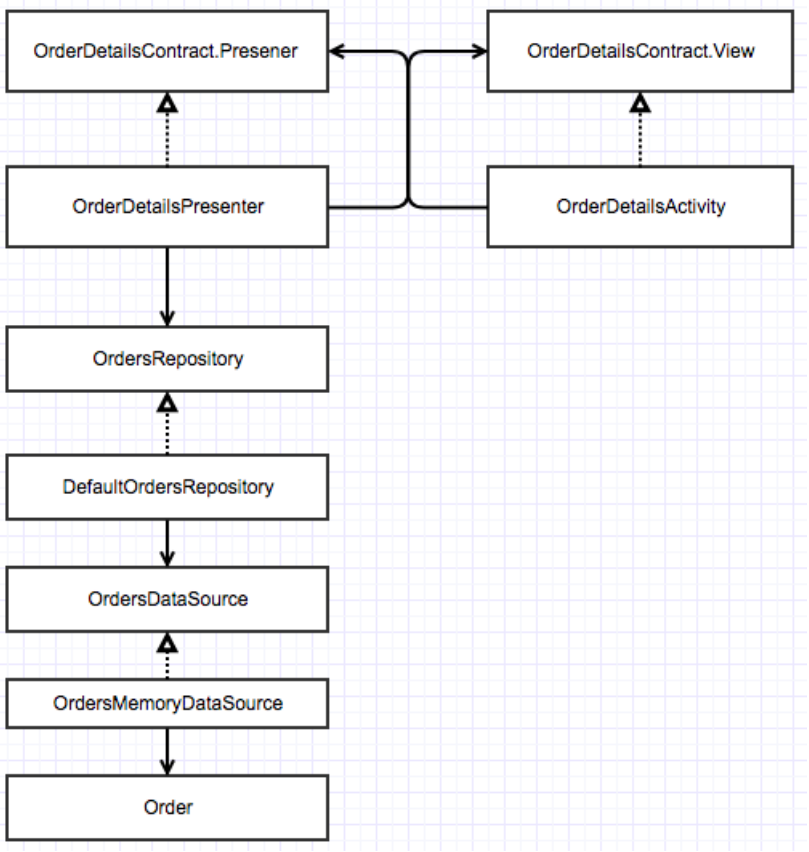


Figura 14 – Exemplo Detalhes do Pedido

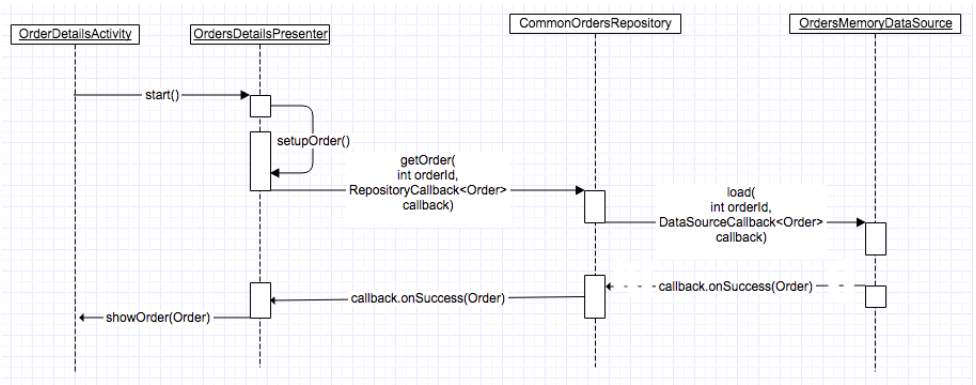


Figura 15 – Detalhes do Pedido Diagrama de Sequencia - Cenário de Sucesso

4.2 Utilização de Padrões de Projetos

No desenvolvimento da aplicação procurou-se utilizar diversos padrões de projeto (23) a seguir será discutido a utilização do padrão Visitor.

4.2.1 Filtros e o Padrão Visitor

Na seção deste trabalho em que foi introduzida a Tela inicial, foi dito que a adição de novos filtros deveria ser leve para a interface. Nesta seção será mostrado passo a passo a criação de um novo filtro para comprovar o que foi dito anteriormente. Ele deverá filtrar os pedidos por período.

4.2.1.1 Passo 1: Criar o filtro

A Listagem 4.1 mostra a implementação de um novo filtro.

Listagem 4.1 – Criação de um novo Filtro

```
@AutoValue
public abstract class DateFilter implements OrderFilter, Parcelable {

    public abstract Date start();

    public abstract Date end();

    @Override
    public boolean accept(OrderVisitor visitor, Order order) {
        return visitor.filter(this, order);
    }

    public static DateFilter create(Date start, Date end) {
        return new AutoValue_DateFilter(start, end);
    }
}
```

Para criar um novo filtro é necessário apenas implementar a interface OrderFilter como na Listagem 4.2.

Listagem 4.2 – Exemplo de OrderFilter

```
boolean accept(OrderVisitor visitor, Order order);
```

A classe também implementa o método Parcelable, que faz com que um objeto dessa classe possa ser transmitido entre Activities.

Foi utilizada a anotação @AutoValue e o método create possuí um retorno que pode parecer estranho. AutoValue (24) é uma biblioteca que torna a criação de Value Objects - isto é, um objeto é igual à outro se seus valores forem iguais, independente se eles apontam para mesma referência - menos verbosa. Além disso essa biblioteca também

facilita a implementação de Parcelable, por isso foi utilizada. Note que a classe implementa a interface Parcelable, mas não sobrescreve nenhum método, isto fica por conta da extensão Parcel (25) do AutoValue.

Portanto tudo que o filtro tem que se preocupar é em guardar os valores de interesse, ou seja, a data de início e de fim do período que se deseja.

4.2.1.2 Passo 2: Adicionar o filtro na interface OrderVisitor

No Passo 1, foi visto que o método accept apenas delega para o visitor se passando como parâmetro. Deve-se adicionar um novo método à interface OrderVisitor como mostra a Listagem 4.3.

Listagem 4.3 – Adicionar método à Interface OrderFilter

```
public interface OrderVisitor {

    boolean filter(IdFilter filter, Order order);

    boolean filter(StatusFilter filter, Order order);

    boolean filter(DateFilter filter, Order order);
}
```

4.2.1.3 Passo 3: Implementar o novo filtro em todos os OrderVisitor

Todo DataSource é um OrderVisitor, portanto se o DataSource fosse um serviço que consome de uma API poderia ser implementado conforme a Listagem 4.4

Listagem 4.4 – Exemplo de método filter

```
    public boolean filter(DateFilter filter, Order order) {
        this.query += "&start=" + filter.start().toString;
        this.query += "&end=" + filter.end().toString;
        return true;
    }
```

4.2.1.4 Passo 4: Adicionar o filtro à interface

Primeiramente deve-se encontrar algum lugar na interface com o usuário para colocar algum botão que ao ser pressionado dispare uma ação de navegar para a Lista de Pedidos passando o novo filtro criado, como pode ser visto na Listagem 4.5.

Listagem 4.5 – Adicionando novo filtro à interface

```
public void onPeriodFilterButtonClicked(
Date start,
Date end) {

    navigateToOrdersList(
            DateFilter.create(start, end),
            false);
}
```

4.2.1.5 Demais explicações

Após a execução desses 4 passos um novo filtro foi criado. Entretanto se houver outros DataSource o novo método filter deverá ser implementado para cada um. E pode haver casos onde a adição de diversos filtros não seja tão simples, como adicionar uma query string.

Entretanto o ponto que se deseja ressaltar aqui é que a interface é livre para criar filtros ou misturá-los, ficando a responsabilidade de como tratá-los com as camadas mais baixas.

Para isso o Padrão Visitor (26) foi de grande ajuda, pois com ele foi possível misturar diversos filtros, com vários Visitors, ou seja vários DataSource. A característica de despacho duplo do padrão permite que o tipo do filtro e de quem vai tratá-lo sejam descobertos sem que se faça testes e cast de tipos explicitamente como mostra a Listagem 4.6:

Listagem 4.6 – Exemplo de conversão de tipo explicita

```
if (filter instanceOf DateFilter )) {
   DateFilter dateFilter = (DateFilter) filter;
   //usa o filtro
}
// ifs com todos os outros filtros
```

4.3 A Camada de Apresentação

Nesta seção será discutida a criação de uma tela. Desde a criação de seu layout, sua Activity utilizando a arquitetura MVC e sua refatoração para MVP. Para tal será utilizada a Tela de Processamento como exemplo.

4.3.1 Criação do Layout

O arquivo de layout é o mesmo para as versões MVC e MVP e pode ser consultado no Anexo B.1.1 o arquivo:

`activity_bacrode_processor.xml`

Como pode ser visto, pela extensão do arquivo, se trata de um arquivo XML. No layout em questão os seguintes elementos são utilizados:

- **Toolbar - Cabeçalho:** Barra superior responsável por comportar o título da tela, que, no caso, é o ID do pedido. Assim como o botão para fechar a câmera e voltar para a tela Detalhes do Pedido.
- **SurfaceView - Preview:** Parte da tela responsável por desenhar a visualização da câmera.
- **View - Guia:** Indicação no meio da tela, para facilitar ao apontar a câmera. Fornece feedback visual quando um item é processo com sucesso ou não
- **TextView - Rodapé:** Barra inferior responsável por mostrar a quantidade de items pendentes de processamento.

4.3.2 Criação da Activity com a arquitetura MVC

O arquivo XML possuí apenas os elementos. Para podemos dar valores a eles, como definir qual será o título da tela, precisamos utilizar uma Activity (ou um Fragment). A Activity que representa a tela de Processamento é a BarcodeProcessorActivity.

Listagem 4.7 – Activity que representa a Tela de Processamento

```
public class BarcodeProcessorActivity
extends AppCompatActivity
implements OnItemProcessedListener
```

Para criar uma Activity basta herdar de AppCompatActivity. A interface OnItemProcessedListener obriga a classe a implementar o método mostrado na Listagem 4.8.

Listagem 4.8 – Interface OnItemProcessedListener

```
void onItemProcessed(String code);
```

Esse método é chamado toda vez que a câmera reconhece um código de barras. Os detalhes de configuração da câmera serão omitidos nessa seção e podem ser encontrados na implementação completa da classe na Listagem B.70.

A partir do momento que a classe herda de uma Activity ela está sujeita ao seu ciclo de vida (27). O primeiro método chamado é o onCreate, um exemplo da implementação desse método aparece na Listagem 4.9.

Listagem 4.9 – Exemplo onCreate

```
@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_barcode_processor);

    // get reference to elements
    //(Header, Preview, Guide e Footer)

    // get CodesToProcess object
    // passed as argument on activity initialization

    // initialize camera and barcode detector

    // initialize toolbar
    // initialize itens left
}
```

Nele é definido qual o arquivo de layout será utilizado e então são criadas referências para os elementos citados. Para o correto funcionamento dessa Activity é esperado que se passe um objeto CodesToProcess como argumento na sua inicialização.

O objeto CodesToProcess possuí um Map com a chave sendo o código a processar e o valor a quantidade de vezes que ele deve ser processado. Esse objeto também contém qual o ID do pedido.

Toda vez que um código novo é detectado o seguinte método da Listagem 4.10 é executado.

Listagem 4.10 – Método onItemProcessed

```
@Override
public void onItemProcessed(String code) {
    if (!ready) {
        return;
    }

    ready = false;

    CodesToProcess.Status status = codesToProcess.process(code);
    showMessage(status, code);

    if (status == CodesToProcess.Status.SUCCESS) {
        showProcessing(R.color.green);
        setItemsLeft();
        checkFinish();
    } else {
        showProcessing(R.color.red);
    }
}
```

Percebe-se que a Activity fica responsável por lidar com muita lógica. Ela é responsável por verificar se o código é válido e tomar as ações correspondentes.

4.3.3 Refatoração da Activity para a arquitetura MVP

O problema da abordagem anterior é que ao testar a lógica dessa Activity deve-se lidar também com os componentes de interface. Para facilitar o teste essa lógica deve ser extraída para uma outra classe e a View ficar responsável apenas por repassar as informações obtidas para essa classe e atualizar a tela. O teste então pode ser feito em cima da classe com a lógica, com um pequeno risco de problemas na apresentação das informações. Esse é um conceito chamado de Passive View por Martin Fowler (7).

4.3.3.1 Extração do Contrato

Conforme mostrado na seção Arquitetura da Aplicação, tanto o Presenter quanto a View são implementações de interfaces. Desacomplando a Activity de um Presenter específico, e o Presenter de uma View específica.

Para realizar este desacoplamento é extraído um contrato que é uma interface que contém os métodos que são responsabilidade da View e os que são responsabilidade do Presenter. Um benefício da criação de um contrato é que apenas olhando o contrato ficam claras as responsabilidades de cada um.

É importante ressaltar que os métodos da View devem ser o mais atômico possível. Por exemplo um método *showMessage* pode ser dividido em *showErroMessage* e *showSuccessMessage*, pois os dois últimos mostram que tem apenas uma responsabilidade, logo não precisam executar lógica para mostrar um tipo ou outro de mensagem.

A Listagem 4.11 mostra uma versão reduzida do contrato estabelecido pela Tela de Processamento, a versão completa pode ser encontrada na Listagem B.71.

Listagem 4.11 – Versão reduzida do BarcodeProcessorContract

```
interface BarcodeProcessorContract {
    interface View extends BaseView<Presenter> {
        void showProcessSuccess();

        void showSuccessMessage(String code);
    }

    interface Presenter extends BasePresenter {
        void onItemProcessed(String code);
    }
}
```

4.3.3.2 Criação da View

Na versão MVP a BarcodeProcessorActivity implementa também BarcodeProcessorContract.View, conforme exposto na Listagem 4.12.

Listagem 4.12 – Implementando BarcodeProcessorContract.View

```
public class BarcodeProcessorActivity
extends AppCompatActivity
```

```
    implements OnItemProcessedListener,
    BarcodeProcessorContract.View
```

Logo os métodos do contrato devem ser implementados. A Listagem 4.13 mostra a implementação deles.

Listagem 4.13 – Implementando os métodos do BarcodeProcessorContract.View

```
    @Override
    public void showProcessSuccess() {
        showProcessing(R.color.green);
    }

    @Override
    public void showSuccessMessage(String code) {
        showMessage(String.format(
                getString(R.string.barcode_processor_success_message),
                code));
}
```

Percebe-se que os métodos não possuem lógica, eles apenas fazem a devida formatação do que deve ser apresentado na tela e apresentam. A responsabilidade por chamar cada método no momento certo fica por conta do Presenter.

Ao receber uma ação - como o clique de um botão pelo usuário, ou, no caso, um código processado pela câmera - a View apenas delega para o Presenter o tratamento da informação, como pode ser visto na Listagem 4.14.

Listagem 4.14 – View delegando para o Presenter

```
@Override
    public void onItemProcessed(String code) {
        presenter.onItemProcessed(code);
    }
```

4.3.3.3 Criação do Presenter

O presenter deve implementar a interface BarcodeProcessorContract.Presenter, como na Listagem 4.15.

Listagem 4.15 – Implementando BarcodeProcessorContract.Presenter

```
class BarcodeProcessorPresenter
implements BarcodeProcessorContract.Presenter
```

Logo os métodos do contrado devem ser implementados. A Listagem 4.16 mostra a implementação deles.

Listagem 4.16 – Implementando os métodos do BarcodeProcessorContract.Presenter

```
    @Override
    public void onItemProcessed(String code) {
    if (!ready) {
        return;
    }
    ready = false;

    CodesToProcess.Status status = codesToProcess.process(code);

    if (status == CodesToProcess.Status.SUCCESS) {
        onProcessSuccess();
    } else {
        view.showProcessError();
    }

    showProcessMessage(status, code);
}

private void showProcessMessage(CodesToProcess.Status status,
String code) {
    switch (status) {
        case SUCCESS:
            view.showSuccessMessage(code);
            break;
        case CODE_ALREADY_PROCESSED:
```

```
                view.showCodeAlreadyProcessedMessage(code);
                break;
            case CODE_INVALID:
                view.showCodeInvalidMessage(code);
                break;
        }
    }
```

Toda a lógica que estava na versão MVC, misturada com elementos de interface, foi transferida para o presenter que é responsável por chamar a view no momento certo com as informações a serem exibidas.

5 Testes

Após o desenvolvimento da aplicação utilizando duas arquiteturas diferentes para a camada de apresentação foram criados testes para as classes pertencentes a ela.

Os seguintes Frameworks foram utilizados:

- **JUnit (28):** para criação de testes de unidade automatizados.
- **Mockito (29):** auxilia na criação de mocks.
- **Roboletric (30):** para criação de testes unitários automatizados para classes que utilizam o Android SDK, como as Activities.

As seções que seguem dão uma visão geral de como foram criados os testes para cada arquitetura. Os testes completos podem ser encontrados no Anexo B.2.1 e Anexo B.3.1.

5.1 Versão MVP

Assim como comentado anteriormente neste trabalho a versão MVP extrai a lógica para um Presenter e a View se torna passiva. Portanto os testes do Presenter testam a lógica e utilizam um Mock da View apenas para verificar se o método correto foi chamado. Em contrapartida os testes da View apenas verificam se as informações são apresentadas corretamente.

5.1.1 Testando o Presenter

A classe OrdersListPresenterTest é detalhada aqui, ela pode ser encontrada, na íntegra, na Listagem B.98.

Primeiramente são declarados os Mocks com a anotação @Mock que faz parte do Framework Mockito, conforme a Listagem 5.1.

Listagem 5.1 – Declaração de Mocks

```
@Mock
private OrdersListContract.View ordersListView;

@Mock
```

```
    private OrdersRepository ordersRepository;

    @Mock
    private OrderFilterList filters;

    @Captor
    private ArgumentCaptor<RepositoryCallback<List<Order>>>
    repositoryCallbackArgumentCaptor;
```

Métodos anotados com @Before provida pelo JUnit rodam antes de cada teste. Para a classe analisada ele apenas inicia os Mocks e o Presenter, injetando nele os Mocks criados. A Lisitagem B.6 mostra essa inicialização.

Listagem 5.2 – Inicialização do teste do Presenter

```
    @Before
    public void setupOrdersListPresenter() {
        MockitoAnnotations.initMocks(this);

        setProcessAllPresenter();
    }

     private void setProcessAllPresenter() {
        ordersListPresenter = new OrdersListPresenter(
                ordersListView,
                filters,
                ordersRepository,
                true);
    }
```

Cada teste deve ser anotado com @Test do JUnit. A Listagem 5.3 mostra o teste de um cenário de sucesso quando o Presenter é iniciado.

Listagem 5.3 – Exemplo de teste do Presenter - caso de sucesso

```
    @Test
    public void onStart_setupOrders_success() {
```

```
        // ORDERS = LISTA COM PEDIDOS

        ordersListPresenter.start();

        verify(ordersRepository).getList(
                eq(filters),
                repositoryCallbackArgumentCaptor.capture());

        repositoryCallbackArgumentCaptor.getValue().onSuccess(ORDERS);

        verify(ordersListView).showOrdersList(ORDERS);
    }
```

Percebe-se como a utilização de Mocks facilita a escrita do teste. Primeiro é chamado o método que se deseja testar na classe em questão, depois é verificado se o Mock do repositório recebeu uma chamada para o método que se espera, após com o uso do repositoryCallbackArgumentCaptor é Mockado uma resposta de sucesso. Finalmente é verificado se a View recebe uma chamada para o método showOrdersList com os pedidos corretos.

Para testar um caso de erro basta criar um Mock com uma resposta de erro com repositoryCallbackArgumentCaptor e verificar se o método showError com a mensagem correta recebe uma chamada, como mostrado na Listagem 5.4.

Listagem 5.4 – Exemplo de teste do Presenter - caso de erro

```
    @Test
    public void onStart_setupOrders_error() {
        // ERRORS = MENSAGENS DE ERRO

        ordersListPresenter.start();

        verify(ordersRepository).getList(
                eq(filters),
                repositoryCallbackArgumentCaptor.capture());

        repositoryCallbackArgumentCaptor.getValue().onError(ERRORS);
        verify(ordersListView).showError(ERRORS.get(0));
    }
```

5.1.2 Testando a View

A lógica já foi testada pelo Presenter, e está garantido que o método certo da View para apresentar cada informação é chamado. Caso se deseje verificar que as informações estão sendo apresentadas de forma correta na tela pode-se testar a View também.A classe OrdersListActivityTest é detalhada aqui, ela pode ser encontrada, na íntegra, na Listagem B.97.

No trecho de código da Listagem 5.5 observa-se que é criado um Mock do Presenter. Nota-se, também, a utlização do Framework Roboletric para poder ter acesso ao Android SDK, como a Activity.

Listagem 5.5 – Inicialização do teste da View

```
@Mock
OrdersListContract.Presenter presenter;

@Before
public void setup() throws Exception {
    MockitoAnnotations.initMocks(this);
    activity = Robolectric.setupActivity(OrdersListActivity.class);

    activity.setPresenter(presenter);
}
```

Como a View é passiva o teste de uma ação do usuário, como apertar um botão, apenas precisa verificar se o método correto do Presenter, que trata essa informação, foi chamado. Conforme mostra a Listagem 5.6.

Listagem 5.6 – Exemplo de teste da View delegando para o Presenter

```
@Test
public void onFabClick() throws Exception {
    activity.onFabClick();
    verify(presenter).onFabCLick();
}
```

A Listagem 5.7 mostra um teste que deseja verificar se as informações são mostradas de forma correta na tela. Primeiramente o método que faz parte do Contrato da

View é chamado com os parâmetros que se deseja apresentar. Em seguida o elemento que se deseja verificar se está apresentando a informação desejada é recuperado. Finalmente verifica-se se o que está sendo apresentado é igual ao desejado.

Listagem 5.7 – Exemplo de teste de View verificando informações na tela

```
@Test
public void showError() throws Exception {
    String error = "Error";
    activity.showError(error);

    EmptyStateAdapter.ViewHolder holder =
            (EmptyStateAdapter.ViewHolder) activity.recyclerView
            .findViewHolderForAdapterPosition(0);

    assertEquals(holder.getMessage().getText().toString(),
    error);
}
```

5.2 Versão MVC

Diferente da versão MVP a lógica da aplicação está misturada com os componentes de interface gráfica. Um teste de um Presenter que testa a lógica e apenas confia a um Mock da View se o método de apresentação correto foi chamado, na versão MVC o teste da mesma lógica deve ser feito, porém a informação que é apresentada deve ser verificada através dos elementos da interface para que se confirme que ela está correta.

5.2.1 Testando a Activity

O mesmo teste mostrado na seção Testando a View que verificava se a mensagem de erro é mostrada corretamente é apresentado na Listagem 5.8 na sua versão MVC.

Listagem 5.8 – Exemplo teste de caso de erro

```
@Test
public void setupOrdersList_error() throws Exception {
    OrdersFakeDataSource.addOrder(ERRORS);

    Intent intent = new Intent();
```

```
        intent.putExtra(
        OrdersListActivity.EXTRA_PROCESS_ALL,
        processAll);

        intent.putExtra(
        OrdersListActivity.EXTRA_ORDER_FILTERS,
        NullOrderFilterList.create());

        activity = Robolectric.buildActivity
        (OrdersListActivity.class,
        intent).create().start().get();

        EmptyStateAdapter.ViewHolder holder =
                (EmptyStateAdapter.ViewHolder)
                activity.recyclerView
                .findViewHolderForAdapterPosition(0);

        assertEquals(
        holder.getMessage().getText().toString(),
        OrdersFakeDataSource.ERROR_MESSAGE);
    }
```

Nota-se que ele faz o mesmo trabalho, porém com mais linhas de código. Isto porque na versão MVC não basta garantir que o método apresenta o que deve apresentar quando chamado. Deve-se garantir também que ele seja chamado na hora certa, ou seja testar se a lógica está correta. No caso do teste em questão ele se encarrega de iniciar a Activity em um estado em que a mensagem de erro deve ser exibida e depois verifica se a informação está sendo exibida corretamente, esta última parte sendo semelhante à versão MVP.

5.2.2 Criando um DataSource Falso

Diferentemente da versão MVP onde o Presenter recebe o repositório pelo seu construtor, facilitando o uso de um Mock, na versão MVC o repositório não é recebido pelo construtor.

Portanto para se testar um cenário de erro de comunicação, por exemplo, o teste deveria rodar em um ambiente onde o DataSource estivesse incomunicável. Forçar uma

situação dessas pode ser difícil, como desligar a internet só para esse teste - caso o DataSource seja um serviço de consumo de API remota.

Para contornar esse problema foram utilizados duas técnicas em conjunto. A principal é a criação de uma classe responsável por fornecer dependências, nesse caso fornecer o repositório correto em ambiente de teste ou produção. A Listagem 5.9 mostra como isso é feito.

Listagem 5.9 – Injeção de Repositório

```
ordersRepository = Inject.provideOrdersRepository();
```

A segunda técnica é a utilização de Flavors (31), ou seja, variantes de compilação onde para cada variante a classe Inject é definida porém cada uma injeta um repositório com um DataSource diferente.

Na configuração de produção um DataSource em memória é injetado, conforme mostrado na Listagem 5.10.

Listagem 5.10 – Injeção com DataSource em Memória

```
public static OrdersRepository provideOrdersRepository() {
    OrdersDataSource ordersDataSource =
    new OrdersMemoryDataSource();
    return new CommonOrdersRepository(ordersDataSource);
}
```

Já na configuração de teste um DataSource que apenas cria um Mock com as respostas é injetado, como pode ser visto na Listagem 5.11.

Listagem 5.11 – Injeção com DataSource Falso

```
public static OrdersRepository provideOrdersRepository() {
    OrdersDataSource ordersDataSource =
    new OrdersFakeDataSource();
    return new CommonOrdersRepository(ordersDataSource);
}
```

Detalhes na implementação do OrdersFakeDataSource pode ser encontrado na Listagem B.40.

6 Aplicação da Métrica de Testabilidade

Após o desenvolvimento da aplicação e dos testes da camada de apresentação foram analisadas métricas baseadas em código fonte - conforme visto na Fundamentação Teórica deste trabalho - afim de avaliar a testabilidade das versões Model-View-Controller e Model-View-Presenter utilizando essa metodologia.

Bruntink (1) define o esforço para se testar uma classe como sendo o tamanho do conjunto de testes. E decide utilizar as seguintes métricas:

- **tLOC - Lines Of Codes:** O número de linhas de código necessários para testar a classe.
- **NOTC - Number of Test Cases:** O número de casos de testes necessários para testar a classe.

Ele busca, ainda, encontrar preditores para estas métricas, baseados em métricas de código fonte. E é demonstrada uma correlação moderadamente forte com as métricas RFC e LOC.

No presente trabalho tanto os preditores encontrados Bruntink (RFC e LOC), quanto as variáveis dependentes (NOTC e tLOC), propostas por ele, foram utilizadas para avaliar a testabilidade das arquiteturas MVC e MVP. Isso foi feito com a finalidade de se encontrar mais de um indicador, no caso de estudo analisado, em favor da testabilidade de uma ou outra arquitetura. Mitigando, assim, possíveis vícios nas escritas dos testes.

Para fazer o levantamento dessas métricas foi utilizado o plugin MetricsReloaded (32) para o Android Studio (33), assim como a ferramenta embutida neste ambiente para aferimento da cobertura de código.

Devido ao fato de que com arquiteturas diferentes algumas partes do código ficam mais difíceis de serem testadas, foi aplicada a métrica de cobertura de código para certificar de que diferentes versões cobrem o código de maneira equivalente.

As métricas RFC e LOC de avaliação da testabilidade de uma classe Java propostas por Bruntink (1) foram levantadas e são apresentadas na Tabela 1. São apresentadas também as métricas levantadas a partir dos códigos dos testes escritos (NOTC e tLOC).

6.1 Análise das métricas de Teste

Conforme dito as métricas NOTC e tLOC foram utilizadas para avaliar o tamanho do conjunto de testes. Analisando a Tabela 1 verifica-se que para a métrica NOTC nas

Tabela 1 – Métrica por tela

	RFC	LOC	NOTC	tLOC	Cobertura
Lista de Pedidos (MVC)	43	166	15	215	91%
Lista de Pedidos (Presenter)	17	91	11	114	93%
Detalhes do Pedido (MVC)	85	323	30	377	85%
Detalhes do Pedido (Presenter)	59	166	33	311	97%
Tela de Processamento (MVC)	61	192	10	98	84%
Tela de Processamento (Presenter)	24	77	10	113	100%

telas *Detalhes do Pedido* e *Tela de Processamento* a versão MVP tem um valor maior na primeira e igual na segunda. Isso se deve ao fato de que para alguns componentes o teste no MVC ser muito difícil de ser realizado utilizando as ferramentas utilizadas. Logo, conseguiu-se fazer mais casos de testes para a arquitetura MVP. Na *Tela de Pedidos* o valor é maior na versão MVC, como o esperado. Embora possa parecer que os números deveriam - se pudesse testar as mesmas coisas nas daus arquituras - ser iguais, visto que eles testam as mesmas funcionalidades, porém para como o MVC está altamente acoplado com o ciclo de vida da Activity, sendo assim um comando de pressionar o botão de voltar do Android ou o botão de fechar da aplicação pode ser mapeado para a mesma ação de *close* no Presenter.

Observando agora a métrica tLOC os valores para a arquitetura MVP são menores, portanto melhores para esta métrica, nas telas *Lista de Pedidos* e *Detalhes do Pedido*. Na *Tela de Processamento* a versão MVP mostra um valor maior para esta métrica. Porém o mesmo comentário, sobre analisar a cobertura de código e o fato de alguns casos de testes não serem possíveis no MVC, valem aqui.

6.2 Análise das métricas de Código Fonte

Conforme comentado na abertura deste capítulo, também foram analisados os preditores propostos por Bruntink (1), a fim de atenuar possíveis vícios nas escritas dos testes e levantar mais dados que possam apontar uma vantagem da arquitetura Model-View-Presenter ou Model-View-Controller no quesito testabilidade.

Ao analisar novamente a Tabela 1, desta vez observando os valores mensurados para as métricas baseadas em código fonte, RFC e LOC, nota-se que ambas apontam para uma melhor testabilidade da variante MVP do código - tendo em vista que, segundo Bruntink (1), quanto menor os valores mais testável é a classe.

6.3 Considerações

Conforme diz Martin Fowler (7) uma View passiva - como é o caso na variante Model-View-Presenter desenvolvida - permite que os testes sejam focados no Presenter com pouco risco de problemas na View. Logo o aumento da testabilidade creditado a esta arquitetura e analisado neste trabalho se dá ao fato de que a View, inerentemente difícil de ser testada - devido à forte dependência da API do Framework de Interface - passa a ter suas responsabildade reduzidas ao mínimo, a ponto de que os testes possam ser apenas do Presenter.

7 Conclusão

Considerando o objetivo proposto por este trabalho - comparar a testabilidade de um aplicativo Android utilizando Model View Controller e Model View Presenter como arquiteturas da camada de apresentação - o presente trabalho se encarregou de alcançá-lo através da criação de um aplicativo Android desenvolvido nas duas variantes a serem comparadas e a escrita de testes de unidade para testá-las. Assim como a posterior comparação destas arquiteturas utilizando tanto métricas baseadas em código fonte quanto métricas baseadas nos próprios testes, ambas propostas por Bruntink (1).

Conforme visto no capítulo Aplicação da Métrica de Testabilidade ambas as métricas - baseadas em código fonte e baseadas em código de testes - apontaram para uma melhor testabilidade da versão Model-View-Presenter sobre a Model-View-Controller, devido ao fato de a primeira variante isolar a lógica de apresentação do código inerentemente difícil de ser testado que envolve a manipulação de componentes de interface que são providos pelo Framework Android.

Como parte do objetivo de comparar a testabilidade das arquiteturas MVC e MVP no desenvolvimento para Android, este trabalho propôs a criação de um aplicativo para ser utilizado como caso de estudo para aplicação das métricas. A aplicação se propôs a auxiliar na montagem de pedidos recebidos por lojas virtuais. Tal aplicação tira proveito das câmeras que equipam os celulares modernos e seu poder de processamento para poder ler códigos de barras de produtos e montar os pedidos de maneira mais confiável. No desenvolvimento atentou-se para que o aplicativo tivesse uma boa usabilidade e isto foi alcançado utilizando o Guia de Estilo proposto pelo Google, assim como levando em consideração as Heurísticas de Nielsen.

Algumas dificuldades foram encontradas durante a realização deste trabalho. Primeiramente, na fase de construção da solução, desenvolver uma boa experiência de uso da aplicação não é uma tarefa fácil e exigiu diversas alterações durante o desenvolvimento. Ainda na fase de construção da aplicação existiu a dificuldade de criar uma arquitetura que facilite modificações futuras. Porém a tarefa mais árdua foi o desenvolvimento dos testes. Testar componentes de interface gráfica muitas vezes se tornou um grande esforço de pesquisa de ferramentas. O Roboletric é uma ferramenta que ajudou bastante neste quesito, no entanto algumas especificidades exigiam um conhecimento mais profundo de seu uso. Pode-se dizer que esta dificuldade, independente de métricas, é um ponto a mais para se testar apenas o Presenter.

Do ponto de vista pessoal este trabalho cumpriu com o objetivo de forçar o autor conhecer mais sobre testes no desenvolvimento Android e aprofundar em questões de

arquitetura testável e escalável na criação de aplicativos para a plataforma.

7.1 Trabalhos Futuros

Os objetivos propostos foram cumpridos e novas ideias de trabalhos surgiram. Um trabalho em Engenharia de Usabilidade que avalie a usabilidade da solução desenvolvida utilizando alguma métrica como SUS (System Usability Scale) e proponha o redesign seria interessante. Outra proposta é avaliar as arquiteturas da camada de apresentação utilizando outras métricas, baseadas em código-fonte ou não. Por exemplo os testes do Presenter rodam mais rápidos do que os que envolvem a View.

Ultimamente outras arquiteturas de camada de apresentação tem ganhado força no desenvolvimento para Android, uma delas é o MVVM. Um trabalho que comparasse a testabilidade de um mesmo aplicativo em versões MVVM e MVP poderia ser relevante.

É muito importante ressaltar que os testes foram realizados utilizando o Roboletric. Existe um outro framework chamado Espresso (34) que pode alterar alguns casos de testes na versão MVP, pois interage de outra maneira com os componentes e algo que era difícil de ser testado, com as ferramentas utilizadas, podem se tornar mais fácil utilizando o Espresso. O fato da ter sido escolhido o Roboletric é que os testes rodam mais rápido, pois não necessitam rodar em um emulador, como é o caso do Espresso. Portanto considero interessante fazer uma comparação entre as versões utilizando este outro framework.

Finalmente um trabalho que propusesse uma outra solução para o mesmo problema - erros na montagem de pedidos - utilizando outra plataforma, por exemplo um site responsivo, e verificasse e comparasse a eficiência das soluções.

Referências

1 BRUNTINK, M. Testability of object-oriented systems: a metrics-based approach. *Universiy Van Amsterdam*, v. 45, 2003. Citado 6 vezes nas páginas 5, 23, 25, 63, 64 e 67.

2 ANDROID. *Android – Google Inc.* 2018. https://developer.android.com/guide/components/fundamentals.html. Acessado: 31/03/2018. Citado na página 26.

3 Android – Google Inc. *Activity*. 2017. https://developer.android.com/reference/android/app/Activity.html. Acessado: 15/07/2017. Citado na página 27.

4 FOWLER, M. *GUI Architectures*. https://martinfowler.com/eaaDev/uiArchs.html. Acessado: 15/07/2017. Citado 2 vezes nas páginas 27 e 28.

5 FOWLER, M. *MVP*. https://martinfowler.com/eaaDev/ModelViewPresenter.html. Acessado: 15/07/2017. Citado na página 27.

6 FOWLER, M. *Supervising Controller*. https://martinfowler.com/eaaDev/SupervisingPresenter.html. Acessado: 15/07/2017. Citado na página 27.

7 FOWLER, M. *Passive View*. https://martinfowler.com/eaaDev/PassiveScreen.html. Acessado: 15/07/2017. Citado 4 vezes nas páginas 27, 28, 49 e 65.

8 FOWLER, M. *Presentation Model*. https://martinfowler.com/eaaDev/PresentationModel.html. Acessado: 15/07/2017. Citado na página 29.

9 MADHVANI, A. Structuring android applications: architectural design patterns for rapid development and bug-free and products. Hogeschool West-Vlaanderen, 2016. Citado 2 vezes nas páginas 29 e 30.

10 Google Inc. *Material Design*. https://material.io/components/android/. Acessado: 23/10/2017. Citado 2 vezes nas páginas 30 e 34.

11 Android – Google Inc. *Design*. https://developer.android.com/design/index.html?hl=pt-br. Acessado: 15/07/2017. Citado na página 30.

12 Google Inc. *Floating Action Buttons*. https://material.io/components/android/catalog/floating-action-button/. Acessado: 23/10/2017. Citado 3 vezes nas páginas 30, 35 e 36.

13 Google Inc. *Cards*. https://material.io/guidelines/components/cards.html. Acessado: 23/10/2017. Citado 2 vezes nas páginas 31 e 35.

14 Google Inc. *Snackbars*. https://material.io/components/android/catalog/snackbar/. Acessado: 23/10/2017. Citado 2 vezes nas páginas 31 e 38.

15 Google Inc. *Collapsing Toolbars*. https://material.io/components/android/catalog/collapsing-toolbar-layout/. Acessado: 23/10/2017. Citado 2 vezes nas páginas 32 e 36.

16 MEDEIROS, A. Estratégias de picking na armazenagem. *Instituto de Logística e Supply Chain. Obtido de http://www.ilos.com.br/web/estrategias-de-picking-na-armazenagem/*, 1999. Citado na página 33.

17 CANAL, E. de S. *Como minimizar os erros do picking*. 2014. http://cio.com.br/tecnologia/2014/11/27/como-minimizar-os-erros-do-picking. Acessado: 06/12/2016. Citado na página 33.

18 Fishbowl Inventory. *Products Pricing*. 2016. https://www.fishbowlinventory.com/products/pricing. Acessado: 06/12/2016. Citado na página 33.

19 NIELSEN, J.; MOLICH, R. Heuristic evaluation of user interfaces. In: ACM. *Proceedings of the SIGCHI conference on Human factors in computing systems*. [S.l.], 1990. p. 249–256. Citado na página 34.

20 WIKIPEDIA. *Dependency inversion princple*. https://en.wikipedia.org/wiki/Dependency_inversion_principle. Acessado: 15/07/2017. Citado na página 41.

21 MARTIN, R. *NO DB*. 2012. https://8thlight.com/blog/uncle-bob/2012/05/15/NODB.html. Acessado: 15/07/2017. Citado na página 42.

22 Android – Google Inc. *NetworkOnMainThreadException*. https://developer.android.com/reference/android/os/NetworkOnMainThreadException.html. Acessado: 15/07/2017. Citado na página 42.

23 GAMMA, E. *Design patterns: elements of reusable object-oriented software*. [S.l.]: Pearson Education India, 1995. Citado na página 43.

24 BOURRILLION, M. K. *AutoValue*. https://github.com/google/auto/tree/master/value. Acessado: 15/07/2017. Citado na página 44.

25 HARTER, R. *AutoValue*. https://github.com/rharter/auto-value-parcel. Acessado: 15/07/2017. Citado na página 45.

26 WIKIPEDIA. *Visitor Pattern*. https://en.wikipedia.org/wiki/Visitor_pattern. Acessado: 15/07/2017. Citado na página 46.

27 Android – Google Inc. *Activity Lifecycle*. https://developer.android.com/guide/components/activities/activity-lifecycle.html. Acessado: 15/07/2017. Citado na página 48.

28 JUNIT. *JUnit*. http://junit.org/junit4/. Acessado: 15/07/2017. Citado na página 55.

29 MOCKITO. *Mockito*. http://site.mockito.org/. Acessado: 15/07/2017. Citado na página 55.

30 ROBOLETRIC. *Roboletric*. http://robolectric.org/. Acessado: 15/07/2017. Citado na página 55.

31 Android – Google Inc. *Flavor*. https://developer.android.com/studio/build/build-variants.html?hl=pt-br. Acessado: 15/07/2017. Citado na página 61.

32 LEIJDEKKERS, B. *MetricsReloaded*. https://plugins.jetbrains.com/plugin/93-metricsreloaded. Acessado: 15/07/2017. Citado na página 63.

33 Android Studio. *Adroid Studio – Google Inc.* 2016. https://developer.android.com/studio/index.html. Acessado: 23/10/2017. Citado na página 63.

34 Android – Google Inc. *Espresso*. https://developer.android.com/training/testing/espresso/. Acessado: 01/07/2018. Citado na página 68.

Anexos

ANEXO A – Artigo

Comparação da Testabilidade das Arquiteturas MVC e MVP na Camada de Apresentação em um aplicativo Android

André Alex Araujo Santos Camargo Pereira[1], **Raul Sidnei Wazlawick**[1]

[1] Departamento de Informática e Estatística – Universidade Federal de Santa Catarina (UFSC)
Caixa Postal 476 – 88.040-900 – Florianópolis – SC – Brazil

Abstract. *The advantages of using an archictecutre thas separates presentation logic from GUI components handling are better code legibility, modularity and improve class testability.*
This paper compared MVC and MVP architectures testability. An metric-based approach was used as discussed in Bruntink (2003) [Bruntink 2003].
It is concluded that if the objective is only to test the presentation logic, taking the risk that some GU component is presented incorrectly, the MVP version presents a better testability metric.

Resumo. *As vantagens de utilizar uma arquitetura que separe a lógica de apresentação e a manipulação de componentes de interface gráfica são desde uma melhor legibilidade e modularidade do código, o que facilita a manutenção, até uma melhora na testabilidade das classes.*
Este trabalho comparou a testabilidade das arquiteturas MVC e MVP. Para essa comparação foi utilizada uma abordagem baseada em métricas de código fonte conforme discutidas em Bruntink (2003) [Bruntink 2003].
Sendo concluído que se o objetivo é apenas se testar a lógica da apresentação, assumindo o risco de que algum componente gráfico seja apresentado de maneira incorreta, a versão MVP apresenta uma métrica de testabilidade melhor.

1. Introdução

Uma tela no desenvolvimento Android tem sua estrutura definida em um arquivo XML e os componentes definidos são referenciados em uma classe que herda de Activity. Ao ser utilizado o padrão MVC uma Activity tem muitas responsabilidades:

- Receber ações do usuário.
- Tratar os dados de entrada.
- Se comunicar com a camada Model para atualizar ou recuperar algum dado.
- Tratar o dado recebido para ser apresentado na tela.
- Atualizar o elemento da View com o novo dado.

Todas essas responsabilidades e o alto acoplamento com o Framework Android tornam uma Activity muito difícil de ser testada. Ao utilizar se utilizar a arquitetura MVP a View, representada pela Activity, se torna passiva, apenas notificando o Presenter sobre as interações do usuário passando os dados necessários e recebendo comandos para atualizar a tela. A lógica do que fazer à partir de uma interação é delegada ao Presenter que não tem acoplamento com o Framework Android.

2. Metodologia

Com o objetivo de comparar a testabilidade das arquiteturas foi criado, como um caso de estudo, um aplicativo Android na versão MVC e então ele foi refactorado para a versão MVP. Testes automatizados foram escritos para as principais classes das duas versões. Após o desenvolvimento foram utilizadas as métricas discutidas em Bruntink (2003) [Bruntink 2003] para comparar a testabilidade das arquiteturas.

2.1. Métricas de Testabilidade

Nas seções à seguir serão discutidos o trabalho de Bruntink e a forma como o presente trabalho o utilizou.

2.1.1. Trabalho de Bruntink

Em [Bruntink 2003] é definido o esforço para se testar uma classe como o tamanho do conjunto de testes. E utiliza as seguintes métricas para avaliar esse tamanho.

- **LOC - Lines Of Code:** O número de linhas de código necessários para testar uma classe.
- **NOTC - Number of Test Cases:**: O número de casos de testes necessários para testar uma classe.

  Então ele tenta responder:

  Os valores das métricas baseadas em código fonte para uma classe tem correlação com o número de linhas de código e o número de casos de testes do conjunto de testes correspondente?

Para isso ele analisou 8 métricas de código fonte nos softwares Apache Ant e DocGen e demonstrou uma correlação moderadamente forte nos dois casos para as seguintes métricas:

- **LOCC - Lines Of Code Per Class:** O número de linhas de código de uma classe.
- **RFC - Response For Class:**: O número de métodos e construtores distintos invocados por uma classe.

2.1.2. O presente trabalho

- Utilizou as métricas que medem o tamanho do conjunto de teste (**LOC e NOTC**), nos testes desenvolvidos, para comparar as duas arquiteturas.
- As métricas *preditoras* desses valores (**LOCC e RFC**) foram utilizadas para contornar possíveis vícios nas escritas do testes e para tentar encontrar maiores evidências em favor de uma ou outra arquitetura.

2.2. Caso de Estudo

O aplicativo desenvolvido tenta resolver o problema no momento da montagem de pedidos de uma loja virtual.

Como Eduardo de Souza Canal [de Souza Canal 2014], escreveu:

> ”A implantação de um Warehouse Management System (WMS) utilizando o módulo automatizado (com leitor de código de barras) atua diretamente nos erros citados acima. Fazendo o uso do o coletor de dados no picking, o sistema valida se mesmo item que está sendo lido é o solicitado, no momento da leitura do código de barras do produto, evitando assim que aconteça a separação de objetos incorretos.”

Portanto o caso de estudo é um aplicativo que utiliza a câmera do smartphone para a leitura do código de barras e processamento de uma lista de pedidos. As figuras 1 e 2 a seguir mostram as telas do software desenvolvido.

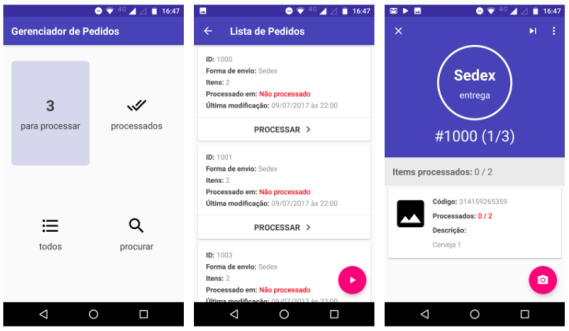


Figura 1. Telas de lista de pedidos

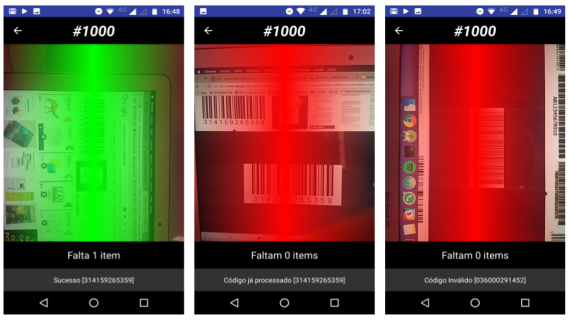


Figura 2. Telas de processamento de pedidos

3. Desenvolvimento

Nesta seção será mostrado como uma Activity foi refactorada da versão MVC para MVP.

Tomando como exemplo na *Tela de Processamento* toda vez que a câmera detecta um código de barras, ela passa o valor lido para a Activity no método mostrado na Listagem 1.

Listagem 1. Lógica de processamento na View na versão MVC

```
@Override
 public void onItemProcessed(String code) {
     if (!ready) {
         return;
     }
```

```
        ready = false;

        CodesToProcess.Status status = codesToProcess.process(code);
        showMessage(status, code);

        if (status == CodesToProcess.Status.SUCCESS) {
            showProcessing(R.color.green);
            setItemsLeft();
            checkFinish();
        } else {
            showProcessing(R.color.red);
        }
    }
```

Nota-se que a Activity verificar se o código lido é válido ou não e decidir o que fazer. Na versão MVP este método fica como mostrado na Listagem 2. Nesta versão a Activity apenas repassa a ação para o Presenter com o dado a ser processado.

Listagem 2. Método onItemProcessed

```
@Override
public void onItemProcessed(String code) {
    presenter.onItemProcessed(code);
}
```

O Presenter então é responsável por processar o código lido e chamar o método exato na View que mostrará o feedback para o usuário. Conforme mostrado na Listagem 3 em caso de código já processado, por exemplo, o Presenter chama o método da **view.showCodeAlreadyProcessedMessage(code)**.

Listagem 3. Movendo a lógica de processamento para o Presenter

```
@Override
public void onItemProcessed(String code) {
if (!ready) {
    return;
}
ready = false;

CodesToProcess.Status status = codesToProcess.process(code);

if (status == CodesToProcess.Status.SUCCESS) {
    onProcessSuccess();
} else {
    view.showProcessError();
}

showProcessMessage(status, code);
}

private void showProcessMessage(CodesToProcess.Status status,
String code) {
```

```
        switch (status) {
            case SUCCESS:
                view.showSuccessMessage(code);
                break;
            case CODE_ALREADY_PROCESSED:
                view.showCodeAlreadyProcessedMessage(code);
                break;
            case CODE_INVALID:
                view.showCodeInvalidMessage(code);
                break;
        }
    }
```

A Listagem 4 mostra como esse método da View é responsável apenas por saber qual mensagem deve ser mostrada em caso de código já processado.

Listagem 4. Exemplo de método da View na versão MVP

```
@Override
public void showCodeAlreadyProcessedMessage(String code) {
    showMessage(String.format(
            getString(R.string.already_processed_message),
            code));
}
```

Após a refactoração percebe-se que a View, representada pela Acvitity no MVP, apenas repassa as ações para o Presenter e pode ser comandada por ele através de métodos bem simples que apenas se encarregam de interagir com os componentes do Framework Android.

4. Testes

Assim como foi falado na seção de Metodologia, testes foram desenvolvidos para as duas arquiteturas analisadas.

Os seguintes Frameworks foram utilizados:

- **JUnit:** para criação de testes unitários automatizados.
- **Mockito:** auxilia na criação de mocks.
- **Roboletric:** para criação de testes unitários automatizados para classes que utilizam o Android SDK, como as Activities.

As Listagens 5 e 6 mostram exemplos de testes nas arquiteturas MVC e MVP respectivamente. Os dois testes verificam se uma mensagem de erro de comunicação com o servidor foi mostrada quando este tipo de problema ocorrer. Nos dois casos com a chamada *init* no primeiro e *startError* no segundo um estado de erro de comunicação é emulado. Uma diferença que se nota ao observar os testes é que na versão MVC é necessário um conhecimento dos componentes Android para se realizar o teste. Já na versão MVP o teste deve apenas verificar se o método da view responsável por mostrar o estado correto recebeu a mensagem correta.

Listagem 5. Exemplo de teste na versão MVC

```
@Test
    public void setupOrdersList_communicationError() throws Exception {
        init(COMMUNICATION_ERROR);

        EmptyStateAdapter.ViewHolder holder =
                (EmptyStateAdapter.ViewHolder) activity.recyclerView
                        .findViewHolderForAdapterPosition(0);

        assertEquals(
                holder.getMessage().getText().toString(),
                activity.getString(R.string.error_communication));
    }
```

Listagem 6. Exemplo de teste na versão MVP

```
Test
    public void onStart_setupOrders_communicationError() {
        startError(ERRORS_EMPTY);

        verify(ordersListView).showCommunicationError();
    }
```

5. Aplicação da Métrica de Testabilidade

Para fazer o levantamento das métricas, apresentadas na Tabela 1, foi utilizado o plugin **MetricsReloaded** para o **Android Studio**, assim como a ferramenta embutida neste ambiente para aferimento da cobertura de código.

Tabela 1. Métrica por tela

	RFC	LOC	NOTC	tLOC	Cobertura
Lista de Pedidos (MVC)	43	166	15	215	91%
Lista de Pedidos (MVP)	17	91	11	114	93%
Detalhes do Pedido (MVC)	85	323	30	377	85%
Detalhes do Pedido (MVP)	59	166	33	311	97%
Tela de Processamento (MVC)	61	192	10	98	84%
Tela de Processamento (MVP)	24	77	10	113	100%

5.1. Dificuldades de testar MVP

Ao analisar os valores da coluna *Cobertura* da Tabela 1, nota-se que para todas as telas analisadas o valor para a versão MVC é inferior ao MVP. Isso se deve ao fato de que, para alguns componentes, é muito difícil testar com o framework de testes utilizado, o Roboletric. Alguns componentes do Android como Snackbar, ou Permissões não puderam ser testados com a ferramenta utilizada.

5.2. Métricas baseadas no conjunto de teste

Conforme dito na Seção 2.1.2 as métricas NOTC e tLOC foram utilizadas para avaliar o tamanho do conjunto de testes. Analisando a Tabela 1 verifica-se que para a métrica NOTC nas telas *Detalhes do Pedido* e *Tela de Processamento* a versão MVP tem um valor maior na primeira e igual na segunda. Isso se deve ao fato de que para alguns componentes o teste no MVC ser muito difícil de ser realizado utilizando as ferramentas utilizadas. Logo, conseguiu-se fazer mais casos de testes para a arquitetura MVP. Na *Tela de Pedidos* o valor é maior na versão MVC, como o esperado. Embora possa parecer que os números deveriam - se pudesse testar as mesmas coisas nas daus arquituras - ser iguais, visto que eles testam as mesmas funcionalidades, porém para como o MVC está altamente acoplado com o ciclo de vida da Activity, sendo assim um comando de pressionar o botão de voltar do Android ou o botão de fechar da aplicação pode ser mapeado para a mesma ação de *close* no Presenter.

Observando agora a métrica tLOC os valores para a arquitetura MVP são menores, portanto melhores para esta métrica, nas telas *Lista de Pedidos* e *Detalhes do Pedido*. Na *Tela de Processamento* a versão MVP mostra um valor maior para esta métrica. Porém o mesmo comentário, sobre analisar a cobertura de código e o fato de alguns casos de testes não serem possíveis no MVC, valem aqui.

5.3. Métricas baseadas em código fonte

Ao analisar novamente a Tabela 1, desta vez observando os valores mensurados para as métricas baseadas em código fonte, RFC e LOC, nota-se que ambas apontam para uma melhor testabilidade da variante MVP do código - tendo em vista que, segundo Bruntink [Bruntink 2003], quanto menor os valores mais testável é a classe.

6. Conclusão

Considerando os resultados versão MVP é apontada como sendo mais testável do que a MVC, pelos seguintes motivos:

- **View** é inerentemente difícil de ser testada - devido à forte dependência da API do Framework de GUI
- Na versão MVP a **View** passa a ter suas responsabildade reduzidas ao mínimo, à ponto de que os testes possam ser apenas do Presenter

O motivo pela versão MVP ter uma melhor testabilidade pode ser resumido pelas palavras de Martin Fowler [Fowler]:

> A perennial problem with building rich client systems is the complication of testing them. Most rich client frameworks were not built with automated testing in mind. Controlling these frameworks programaticly is often very difficult.
>
> A Passive View handles this by reducing the behavior of the UI components to the absolute minimum by using a controller that not just handles responses to user events, but also does all the updating of the view. This allows testing to be focused on the controller with little risk of problems in the view.

7. Trabalhos Futuros

Como trabalhos futuros são sugeridos:

- Neste trabalho foi utilizado o Framework Roboletric, seria interessante fazer a mesma comparação entre as arquiteturas utilizando o Framework Espresso.
- No Google I/O de 2017 foi lançado o Android Architecture Components que utiliza a arquitetura MVVM. Desenvolver uma terceira versão do aplicativo construído neste trabalho utilizando esta arquitetura e e comparar com a versão MVP.

Referências

Bruntink, M. (2003). Testability of object-oriented systems: a metrics-based approach. *Universiy Van Amsterdam*, 45.

de Souza Canal, E. (2014). Como minimizar os erros do picking. `http://cio.com.br/tecnologia/2014/11/27/como-minimizar-os-erros-do-picking`. Acessado: 06/12/2016.

Fowler, M. Passive view. `https://martinfowler.com/eaaDev/PassiveScreen.html`. Acessado: 15/07/2017.

ANEXO B – Código Fonte

B.1 Código Comum entre as versões MVC e MVP

Listagem B.1 – AndroidManifest

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.com.aaascp.gerenciadordepedidos">

    <uses-permission android:name="android.permission.CAMERA" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="@string/app_name"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">

        <activity
            android:name=".presentation.ui.main.MainActivity"
            android:screenOrientation="portrait">

            <intent-filter>
                <category android:name="android.intent.category.LAUNCHER" />
                <action android:name="android.intent.action.MAIN" />
            </intent-filter>
        </activity>

        <activity
            android:name=".presentation.ui.order.list.OrdersListActivity"
            android:screenOrientation="portrait" />

        <activity
            android:name=".presentation.ui.order.details.OrderDetailsActivity"
            android:screenOrientation="portrait" />

        <activity
            android:name=".presentation.ui.order.info.OrderInfoActivity"
            android:screenOrientation="portrait" />

        <activity
            android:name=".presentation.ui.camera.BarcodeProcessorActivity"
            android:screenOrientation="portrait" />

        <activity
            android:name=".presentation.ui.order.item.OrderItemActivity"
            android:screenOrientation="portrait"
            android:theme="@style/Dialog" />

    </application>
```

```
</manifest>
```

Listagem B.2 – BaseActivity

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui;

import android.support.v7.app.AppCompatActivity;

public abstract class BaseActivity extends AppCompatActivity {}
```

Listagem B.3 – BaseFragment

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui;


import android.app.Fragment;

/**
 * Created by andre on 18/09/17.
 */

public abstract class BaseFragment extends Fragment {
}
```

Listagem B.4 – Empty State Adapter

```
package br.com.aaascp.gerenciadordepedidos.presentation.util;

import android.content.Context;
import android.graphics.drawable.Drawable;
import android.support.annotation.DrawableRes;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import br.com.aaascp.gerenciadordepedidos.R;
import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by andre on 28/09/17.
 */

public final class EmptyStateAdapter extends RecyclerView.Adapter<EmptyStateAdapter.ViewHolder> {

    private final static int NULL_DRAWABLE = -1;
    @DrawableRes
    private final int image;

    private final String message;
    private final LayoutInflater inflater;

    public EmptyStateAdapter(
            Context context,
            @DrawableRes int image,
```

```
            String message) {

        this.image = image;
        this.message = message;

        inflater = LayoutInflater.from(context);
    }

    public EmptyStateAdapter(
            Context context,
            String message) {

        this.image = NULL_DRAWABLE;
        this.message = message;

        inflater = LayoutInflater.from(context);
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        return new ViewHolder(
                inflater.inflate(
                        R.layout.row_empty_state,
                        parent,
                        false));
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        if (image != NULL_DRAWABLE) {
            holder.image.setImageResource(image);
        }

        holder.message.setText(message);
    }

    @Override
    public int getItemCount() {
        return 1;
    }

    public static class ViewHolder extends RecyclerView.ViewHolder {

        @BindView(R.id.empty_state_image)
        ImageView image;

        @BindView(R.id.empty_state_message)
        TextView message;

        ViewHolder(View itemView) {
            super(itemView);

            ButterKnife.bind(this, itemView);
        }

        public ImageView getImage() {
            return image;
        }

        public TextView getMessage() {
```

```
            return message;
        }
    }
}
```

Listagem B.5 – Inject Prod

```
package br.com.aaascp.gerenciadordepedidos;

import br.com.aaascp.gerenciadordepedidos.repository.CommonOrdersRepository;
import br.com.aaascp.gerenciadordepedidos.repository.OrdersRepository;
import br.com.aaascp.gerenciadordepedidos.repository.dao.OrdersDataSource;
import br.com.aaascp.gerenciadordepedidos.repository.dao.OrdersMemoryDataSource;

/**
 * Created by andre on 03/10/17.
 */

public class Inject {

    public static OrdersRepository provideOrdersRepository() {
        OrdersDataSource ordersDataSource = new OrdersMemoryDataSource();
        return new CommonOrdersRepository(ordersDataSource);
    }
}
```

Listagem B.6 – Inject Mock

```
package br.com.aaascp.gerenciadordepedidos;

import br.com.aaascp.gerenciadordepedidos.repository.CommonOrdersRepository;
import br.com.aaascp.gerenciadordepedidos.repository.OrdersRepository;
import br.com.aaascp.gerenciadordepedidos.repository.dao.OrdersDataSource;
import br.com.aaascp.gerenciadordepedidos.repository.dao.OrdersFakeDataSource;
import br.com.aaascp.gerenciadordepedidos.repository.dao.OrdersMemoryDataSource;

/**
 * Created by andre on 03/10/17.
 */

public class Inject {

    public static OrdersRepository provideOrdersRepository() {
        OrdersDataSource ordersDataSource = new OrdersFakeDataSource();
        return new CommonOrdersRepository(ordersDataSource);
    }
}
```

Listagem B.7 – ValueLabelView

```
package br.com.aaascp.gerenciadordepedidos.presentation.custom;

import android.content.Context;
import android.content.res.TypedArray;
import android.graphics.drawable.Drawable;
import android.support.annotation.Nullable;
import android.support.constraint.ConstraintLayout;
import android.support.v4.content.ContextCompat;
import android.util.AttributeSet;
```

```
import android.view.LayoutInflater;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import br.com.aaascp.gerenciadordepedidos.R;
import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by andre on 18/09/17.
 */

public class ValueLabelView extends ConstraintLayout {

    @BindView(R.id.value_label_view_icon)
    ImageView iconView;

    @BindView(R.id.value_label_view_value)
    TextView valueView;

    @BindView(R.id.value_label_view_label)
    TextView labelView;

    private String value;
    private String label;
    private Drawable icon;
    private Drawable background;
    private int color;

    public ValueLabelView(Context context, @Nullable AttributeSet attrs) {
        super(context, attrs);

        extractAttributes(context, attrs);

        LayoutInflater inflater = (LayoutInflater) context
                .getSystemService(Context.LAYOUT_INFLATER_SERVICE);

        View view = inflater.inflate(R.layout.view_value_label, this, true);

        ButterKnife.bind(this, view);

        setAttributes();
    }

    public void setValue(String value) {
        valueView.setText(value);
    }

    private void extractAttributes(Context context, AttributeSet attrs) {
        TypedArray a = context.getTheme().obtainStyledAttributes(
                attrs,
                R.styleable.ValueLabelView,
                0, 0);

        try {
            value = a.getString(R.styleable.ValueLabelView_value);
            label = a.getString(R.styleable.ValueLabelView_label);
            icon = a.getDrawable(R.styleable.ValueLabelView_icon);
            background = a.getDrawable(R.styleable.ValueLabelView_backgroundImage);
```

```
            color = a.getColor(
                    R.styleable.ValueLabelView_color,
                    ContextCompat.getColor(
                            context,
                            R.color.primary_text));

        } finally {
            a.recycle();
        }
    }

    private void setAttributes() {
        setValue();
        setBackground();
        labelView.setText(label);
        labelView.setTextColor(color);
        valueView.setTextColor(color);
    }

    private void setValue() {
        if (icon != null) {
            iconView.setImageDrawable(icon);
            iconView.setVisibility(VISIBLE);
            valueView.setVisibility(GONE);
        } else {
            valueView.setText(value);
        }
    }

    private void setBackground() {
        if (background != null) {
            setBackground(background);
        }
    }

    public String getValue() {
        return valueView.getText().toString();
    }
}
```

Listagem B.8 – App Build Gradle

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 25
    buildToolsVersion "25.0.3"
    defaultConfig {
        applicationId "br.com.aaascp.gerenciadordepedidos"
        minSdkVersion 21
        targetSdkVersion 25
        versionCode 1
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
```

```
        debug {
            // Run code coverage reports by default on debug builds.
            // testCoverageEnabled = true
        }
    }

    buildTypes {

    }

    productFlavors {
        mock {
            applicationIdSuffix = ".mock"
        }

        prod {

        }
    }

    android.variantFilter { variant ->
        if (variant.buildType.name == 'release'
                && variant.getFlavors().get(0).name == 'mock') {
            variant.setIgnore(true);
        }
    }

    // Always show the result of every unit test, even if it passes.
    testOptions.unitTests.all {
        testLogging {
            events 'passed', 'skipped', 'failed', 'standardOut', 'standardError'
        }
    }

    configurations.all {
        resolutionStrategy {
            force 'com.android.support:support-annotations:23.0.1'
        }
    }
}

ext {
    supportLibraryVersion = '25.4.0'
    playServicesVersion = '11.0.4'

    junitVersion = '4.12'
    mockitoVersion = '1.10.19'
    powerMockito = '1.6.2'
    hamcrestVersion = '1.3'
    runnerVersion = '1.0.0'
    rulesVersion = '1.0.0'
    espressoVersion = '3.0.0'
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])

    compile "com.android.support.constraint:constraint-layout:1.0.2"
    compile "com.android.support:appcompat-v7:$supportLibraryVersion"
```

```
    compile "com.android.support:design:$supportLibraryVersion"
    compile "com.android.support:cardview-v7:$supportLibraryVersion"
    compile "com.android.support:recyclerview-v7:$supportLibraryVersion"

    // Network & image loading
    compile 'com.squareup.retrofit2:retrofit:2.3.0'
    compile 'com.squareup.retrofit2:converter-gson:2.3.0'
    compile 'com.squareup.picasso:picasso:2.5.2'

    // Code generators
    compile 'com.jakewharton:butterknife:8.7.0'
    annotationProcessor 'com.jakewharton:butterknife-compiler:8.7.0'

    provided 'com.google.auto.value:auto-value:1.3'
    annotationProcessor 'com.google.auto.value:auto-value:1.3'

    provided 'com.ryanharter.auto.value:auto-value-gson:0.5.0'
    annotationProcessor 'com.ryanharter.auto.value:auto-value-gson:0.5.0'
    annotationProcessor 'com.ryanharter.auto.value:auto-value-parcel:0.2.5'
    annotationProcessor 'com.gabrielittner.auto.value:auto-value-with:1.0.0'

    // Play services
    compile "com.google.android.gms:play-services-vision:$playServicesVersion"

    // Dependencies for local unit tests
    testCompile "junit:junit:$junitVersion"
    testCompile "org.mockito:mockito-all:$mockitoVersion"
    testCompile "org.hamcrest:hamcrest-all:$hamcrestVersion"
    testCompile "org.robolectric:robolectric:3.4.2"
    testCompile "org.powermock:powermock-module-junit4:1.6.4"
    testCompile "org.powermock:powermock-module-junit4-rule:1.6.4"
    testCompile "org.powermock:powermock-api-mockito:1.6.4"
    testCompile "org.powermock:powermock-classloading-xstream:1.6.4"


    // Espresso UI Testing dependencies.
    androidTestCompile "com.android.support.test.espresso:espresso-core:$espressoVersion"
    androidTestCompile "com.android.support.test:rules:$rulesVersion"
    androidTestCompile"com.android.support.test.espresso:espresso-contrib:$espressoVersion"
    androidTestCompile "com.android.support.test.espresso:espresso-intents:$espressoVersion"
}
```

Listagem B.9 – Project Build Gradle

```
// Top-level build file where you can add configuration options common to all sub-projects/modules.

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.3'

        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}

allprojects {
    repositories {
```

```
        jcenter()
        maven {
            url "https://maven.google.com"
        }
    }
}

task clean(type: Delete) {
    delete rootProject.buildDir
}
```

B.1.1 Layouts

Listagem B.10 – Barcode Processor Layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/barcode_processor_root"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:animateLayoutChanges="true"
    android:background="@android:color/black">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <android.support.v7.widget.Toolbar
            android:id="@+id/barcode_processor_toolbar"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="@android:color/black">

            <TextView
                android:id="@+id/barcode_processor_title"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:textColor="@android:color/white"
                android:textSize="@dimen/large_title_text_size"
                android:textStyle="bold|italic"
                tools:text="#1000" />
        </android.support.v7.widget.Toolbar>

    </android.support.design.widget.AppBarLayout>

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        app:layout_dodgeInsetEdges="bottom">

        <SurfaceView
            android:id="@+id/barcode_processor_preview"
            android:layout_width="0dp"
            android:layout_height="0dp"
            app:layout_constraintBottom_toBottomOf="parent"
```

```
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <View
            android:id="@+id/barcode_processor_guide"
            android:layout_width="wrap_content"
            android:layout_height="0dp"
            android:background="@drawable/gradient_horizontal_white_transparent"
            android:rotation="0"
            app:layout_constraintBottom_toTopOf="@+id/barcode_processor_items_left"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="parent" />

        <TextView
            android:id="@+id/barcode_processor_items_left"
            android:layout_width="0dp"
            android:layout_height="@dimen/default_bar_height"
            android:layout_gravity="bottom"
            android:background="@android:color/black"
            android:gravity="center"
            android:textColor="@android:color/white"
            android:textSize="@dimen/title_text_size"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toRightOf="parent"
            tools:text="Faltam␣4␣items" />

    </android.support.constraint.ConstraintLayout>

</android.support.design.widget.CoordinatorLayout>
```

Listagem B.11 – Main Layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="br.com.aaascp.gerenciadordepedidos.presentation.ui.main.MainActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <android.support.v7.widget.Toolbar
            android:id="@+id/main_toolbar"
            style="@style/ToolbarStyle"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            android:background="?attr/colorPrimary"
            app:title="@string/app_name" />

    </android.support.design.widget.AppBarLayout>

    <fragment
        android:id="@+id/main_dashboard"
        android:name="br.com.aaascp.gerenciadordepedidos.presentation.ui.main.dashboard.DashboardFragment"
```

```
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior" />

</android.support.design.widget.CoordinatorLayout>
```

Listagem B.12 – Order Details Layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <android.support.design.widget.CollapsingToolbarLayout
            android:layout_width="match_parent"
            android:layout_height="280dp"
            app:collapsedTitleTextAppearance="@style/Toolbar.TextAppearance.Highlighted"
            app:expandedTitleGravity="bottom|center_horizontal"
            app:expandedTitleTextAppearance="@style/Toolbar.TextAppearance.Expanded"
            app:layout_scrollFlags="scroll|exitUntilCollapsed|snap">

            <br.com.aaascp.gerenciadordepedidos.presentation.custom.ValueLabelView
                android:id="@+id/order_details_ship_type"
                android:layout_width="150dp"
                android:layout_height="150dp"
                android:layout_gravity="center_horizontal"
                android:layout_marginTop="?attr/actionBarSize"
                app:backgroundImage="@drawable/circle_transparent_white_border_4dp"
                app:color="@android:color/white"
                app:label="entrega" />

            <android.support.v7.widget.Toolbar
                android:id="@+id/order_details_toolbar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                app:layout_collapseMode="pin" />

        </android.support.design.widget.CollapsingToolbarLayout>

        <include
            layout="@layout/include_processed_items"
            android:visibility="gone"
            tools:visibility="visible" />

        <include
            layout="@layout/include_processed_finish"
            android:visibility="gone"
            tools:visibility="visible" />

        <include
            layout="@layout/include_processed_already"
            android:visibility="gone"
            tools:visibility="visible" />
```

```
    </android.support.design.widget.AppBarLayout>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/order_details_recycler"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layoutManager="LinearLayoutManager"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        tools:listitem="@layout/row_order_items" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/order_details_fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|bottom"
        android:layout_marginBottom="@dimen/fab_margin"
        android:layout_marginRight="@dimen/fab_margin"
        android:src="@drawable/ic_camera_white_vector" />

</android.support.design.widget.CoordinatorLayout>
```

Listagem B.13 – Order Info Layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/android
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <android.support.v7.widget.Toolbar
            android:id="@+id/order_info_toolbar"
            style="@style/ToolbarStyle"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            app:layout_collapseMode="pin" />

    </android.support.design.widget.AppBarLayout>

    <android.support.v4.widget.NestedScrollView
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_behavior="@string/appbar_scrolling_view_behavior">

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical">

            <include layout="@layout/include_info_order" />

            <include layout="@layout/include_info_shipment" />

            <include layout="@layout/include_info_customer" />
        </LinearLayout>

    </android.support.v4.widget.NestedScrollView>
```

```
</android.support.design.widget.CoordinatorLayout>
```

Listagem B.14 – Order Item Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <android.support.v7.widget.Toolbar
            android:id="@+id/order_item_toolbar"
            style="@style/ToolbarStyle.Small"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize" />

    </android.support.design.widget.AppBarLayout>

    <ScrollView
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_weight="1">

        <ImageView
            android:id="@+id/order_item_image"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:adjustViewBounds="true"
            app:srcCompat="@drawable/ic_image_black_vector" />
    </ScrollView>

    <TextView
        android:id="@+id/order_item_description"
        android:layout_width="match_parent"
        android:layout_height="70dp"
        android:background="@color/light_grey"
        android:ellipsize="end"
        android:gravity="center"
        android:maxLines="3"
        android:padding="8dp"
        android:textSize="@dimen/large_text_size"
        tools:text="Descri   o␣do␣item" />
</LinearLayout>
```

Listagem B.15 – Orders List Layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout xmlns:android="http://schemas.android.com/apk/res/a
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
```

```
    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <android.support.v7.widget.Toolbar
            android:id="@+id/orders_list_toolbar"
            style="@style/ToolbarStyle"
            android:layout_width="match_parent"
            android:layout_height="?attr/actionBarSize"
            app:title="@string/orders_list_title" />

    </android.support.design.widget.AppBarLayout>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/orders_list_recycler"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layoutManager="LinearLayoutManager"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        tools:listitem="@layout/row_orders_list" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/orders_list_fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="right|bottom"
        android:layout_marginBottom="@dimen/fab_margin"
        android:layout_marginRight="@dimen/fab_margin"
        android:src="@drawable/ic_play_white_vector"
        android:visibility="gone"
        tools:visibility="visible" />

</android.support.design.widget.CoordinatorLayout>
```

Listagem B.16 – Dashboard Layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <br.com.aaascp.gerenciadordepedidos.presentation.custom.ValueLabelView
        android:id="@+id/dashboard_to_process"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginBottom="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="32dp"
        app:backgroundImage="@drawable/square_primary_8dp"
        app:label="@string/dashboard_orders_to_process"
        app:layout_constraintBottom_toTopOf="@+id/dashboard_all"
        app:layout_constraintHorizontal_chainStyle="spread"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/dashboard_processed"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_chainStyle="spread" />

    <br.com.aaascp.gerenciadordepedidos.presentation.custom.ValueLabelView
```

```
        android:id="@+id/dashboard_processed"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="16dp"
        app:icon="@drawable/ic_done_all_black_vector"
        app:label="@string/dashboard_orders_processed"
        app:layout_constraintBottom_toBottomOf="@id/dashboard_to_process"
        app:layout_constraintLeft_toRightOf="@+id/dashboard_to_process"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="@+id/dashboard_to_process" />

    <br.com.aaascp.gerenciadordepedidos.presentation.custom.ValueLabelView
        android:id="@+id/dashboard_all"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginBottom="32dp"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="16dp"
        app:icon="@drawable/ic_list_black_vector"
        app:label="@string/dashboard_orders_all"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintHorizontal_chainStyle="spread"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toLeftOf="@+id/dashboard_find"
        app:layout_constraintTop_toBottomOf="@+id/dashboard_to_process" />

    <br.com.aaascp.gerenciadordepedidos.presentation.custom.ValueLabelView
        android:id="@+id/dashboard_find"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="16dp"
        app:icon="@drawable/ic_search_black_vector"
        app:label="@string/dashboard_orders_find"
        app:layout_constraintBottom_toBottomOf="@id/dashboard_all"
        app:layout_constraintLeft_toRightOf="@id/dashboard_all"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="@id/dashboard_all" />

</android.support.constraint.ConstraintLayout>
```

Listagem B.17 – Info Customer Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="@dimen/default_bar_height"
        android:layout_gravity="center_vertical"
        android:gravity="center_vertical"
        android:paddingLeft="8dp"
        android:text="@string/info_customer_title"
```

```
        android:textSize="@dimen/large_text_size"
        android:textStyle="bold"
        tools:ignore="RtlSymmetry" />

    <android.support.v7.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="4dp">

        <android.support.constraint.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <TextView
                android:id="@+id/info_customer_id_label"
                style="@style/OrdersList.Labels"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="8dp"
                android:layout_marginTop="8dp"
                android:text="@string/info_customer_id_label"
                app:layout_constraintBottom_toTopOf="@+id/info_customer_name_label"
                app:layout_constraintLeft_toLeftOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <TextView
                android:id="@+id/info_customer_name_label"
                style="@style/OrdersList.Labels"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:text="@string/info_customer_name_label"
                app:layout_constraintBottom_toTopOf="@+id/info_customer_name_value"
                app:layout_constraintLeft_toLeftOf="@+id/info_customer_id_label"
                app:layout_constraintTop_toBottomOf="@+id/info_customer_id_label" />

            <TextView
                android:id="@+id/info_customer_id_value"
                style="@style/OrdersList.Values"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="4dp"
                app:layout_constraintLeft_toRightOf="@+id/info_customer_id_label"
                app:layout_constraintTop_toTopOf="@+id/info_customer_id_label"
                tools:text="10001" />

            <TextView
                android:id="@+id/info_customer_name_value"
                style="@style/OrdersList.Values"
                android:layout_width="0dp"
                android:layout_height="wrap_content"
                android:layout_marginBottom="8dp"
                android:layout_marginLeft="4dp"
                android:layout_marginRight="8dp"
                android:ellipsize="end"
                android:maxLines="2"
                app:layout_constraintBottom_toBottomOf="parent"
```

```
                app:layout_constraintLeft_toLeftOf="@+id/info_customer_id_label"
                app:layout_constraintRight_toRightOf="parent"
                app:layout_constraintTop_toBottomOf="@+id/info_customer_name_label"
                tools:text="Andre␣Alex␣Araujo␣Santos␣Camargo␣Pereira" />

        </android.support.constraint.ConstraintLayout>
    </android.support.v7.widget.CardView>
</LinearLayout>
```

Listagem B.18 – Info Order Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="@dimen/default_bar_height"
        android:layout_gravity="center_vertical"
        android:gravity="center_vertical"
        android:paddingLeft="8dp"
        android:text="@string/info_order_title"
        android:textSize="@dimen/large_text_size"
        android:textStyle="bold"
        tools:ignore="RtlSymmetry" />

    <android.support.v7.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="4dp">

        <android.support.constraint.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <TextView
                android:id="@+id/info_order_size_label"
                style="@style/OrdersList.Labels"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="8dp"
                android:layout_marginTop="8dp"
                android:text="@string/info_order_size_label"
                app:layout_constraintBottom_toTopOf="@+id/info_order_processed_at_label"
                app:layout_constraintLeft_toLeftOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <TextView
                android:id="@+id/info_order_processed_at_label"
                style="@style/OrdersList.Labels"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
```

```
            android:text="@string/info_order_processed_at_label"
            app:layout_constraintBottom_toTopOf="@+id/info_order_last_modification_label"
            app:layout_constraintLeft_toLeftOf="@+id/info_order_size_label"
            app:layout_constraintTop_toBottomOf="@+id/info_order_size_label" />

        <TextView
            android:id="@+id/info_order_last_modification_label"
            style="@style/OrdersList.Labels"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:layout_marginBottom="8dp"
            android:text="@string/info_order_last_modification_label"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="@+id/info_order_size_label"
            app:layout_constraintTop_toBottomOf="@+id/info_order_processed_at_label" />

        <TextView
            android:id="@+id/info_order_size_value"
            style="@style/OrdersList.Values"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="4dp"
            app:layout_constraintLeft_toRightOf="@+id/info_order_size_label"
            app:layout_constraintTop_toTopOf="@+id/info_order_size_label"
            tools:text="5" />

        <TextView
            android:id="@+id/info_order_processed_at_value"
            style="@style/OrdersList.Values"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="4dp"
            android:layout_marginRight="8dp"
            android:ellipsize="end"
            android:maxLines="2"
            app:layout_constraintLeft_toRightOf="@+id/info_order_processed_at_label"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="@+id/info_order_processed_at_label"
            tools:text="09/07/2017␣ s ␣22:00" />

        <TextView
            android:id="@+id/info_order_last_modification_value"
            style="@style/OrdersList.Values"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginLeft="4dp"
            android:layout_marginRight="8dp"
            android:ellipsize="end"
            android:maxLines="2"
            app:layout_constraintLeft_toRightOf="@+id/info_order_last_modification_label"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toTopOf="@+id/info_order_last_modification_label"
            tools:text="09/07/2017␣ s ␣22:00" />
    </android.support.constraint.ConstraintLayout>
</android.support.v7.widget.CardView>
</LinearLayout>
```

Listagem B.19 – Info Shipment Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:layout_width="match_parent"
        android:layout_height="@dimen/default_bar_height"
        android:layout_gravity="center_vertical"
        android:gravity="center_vertical"
        android:paddingLeft="8dp"
        android:text="@string/info_shipment_title"
        android:textSize="@dimen/large_text_size"
        android:textStyle="bold"
        tools:ignore="RtlSymmetry" />

    <android.support.v7.widget.CardView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_marginBottom="4dp"
        android:layout_marginLeft="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginTop="4dp">

        <android.support.constraint.ConstraintLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content">

            <TextView
                android:id="@+id/info_shipment_type_label"
                style="@style/OrdersList.Labels"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="8dp"
                android:layout_marginTop="8dp"
                android:text="@string/info_shipment_type_label"
                app:layout_constraintBottom_toTopOf="@+id/info_shipment_address_label"
                app:layout_constraintLeft_toLeftOf="parent"
                app:layout_constraintTop_toTopOf="parent" />

            <TextView
                android:id="@+id/info_shipment_address_label"
                style="@style/OrdersList.Labels"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginTop="8dp"
                android:text="@string/info_shipment_address_label"
                app:layout_constraintBottom_toTopOf="@+id/info_shipment_address_value"
                app:layout_constraintLeft_toLeftOf="@+id/info_shipment_type_label"
                app:layout_constraintTop_toBottomOf="@+id/info_shipment_type_label" />

            <TextView
                android:id="@+id/info_shipment_type_value"
                style="@style/OrdersList.Values"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_marginLeft="4dp"
```

```
            app:layout_constraintLeft_toRightOf="@+id/info_shipment_type_label"
            app:layout_constraintTop_toTopOf="@+id/info_shipment_type_label"
            tools:text="Transportadora" />

        <TextView
            android:id="@+id/info_shipment_address_value"
            style="@style/OrdersList.Values"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp"
            android:layout_marginLeft="4dp"
            android:layout_marginRight="8dp"
            android:ellipsize="end"
            android:maxLines="3"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="@+id/info_shipment_type_label"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/info_shipment_address_label"
            tools:text="Avenida Engenheiro Max de Souza, 1293, Coqueiros" />


    </android.support.constraint.ConstraintLayout>
</android.support.v7.widget.CardView>
</LinearLayout>
```

Listagem B.20 – Order Details Header Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="horizontal">




</LinearLayout>
```

Listagem B.21 – Processed Already Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/order_details_processed_root"
    android:layout_width="match_parent"
    android:layout_height="@dimen/default_bar_height"
    android:background="@color/light_grey"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="16dp"
        android:text="@string/order_details_finished"
        android:textSize="@dimen/large_text_size"
        android:textStyle="bold" />

    <Button
```

```
        android:id="@+id/order_details_processed"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:text="@string/order_details_processed_action" />

</LinearLayout>
```

Listagem B.22 – Processed Finish Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/order_details_finish_root"
    android:layout_width="match_parent"
    android:layout_height="@dimen/default_bar_height"
    android:background="@color/light_grey"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="16dp"
        android:text="@string/order_details_finished"
        android:textSize="@dimen/large_text_size"
        android:textStyle="bold" />

    <Button
        android:id="@+id/order_details_finish"
        style="@style/Widget.AppCompat.Button.Borderless"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:text="@string/order_details_finished_action" />

</LinearLayout>
```

Listagem B.23 – Processed Items Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/order_details_items_left_root"
    android:layout_width="match_parent"
    android:layout_height="@dimen/default_bar_height"
    android:background="@color/light_grey"
    android:orientation="horizontal">

    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="16dp"
        android:text="@string/order_details_count"
        android:textSize="@dimen/large_text_size"
        android:textStyle="bold" />
```

```
    <TextView
        android:id="@+id/order_details_items_left"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center_vertical"
        android:layout_marginLeft="4dp"
        android:textSize="@dimen/large_text_size"
        tools:text="1/5" />
</LinearLayout>
```

Listagem B.24 – Cards Empty State Layout

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:gravity="center"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/empty_state_image"
        android:layout_width="104dp"
        android:layout_height="104dp"
        app:srcCompat="@drawable/ic_mood_bad_black_vector" />

    <TextView
        android:id="@+id/empty_state_message"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:textSize="@dimen/large_text_size"
        tools:text="Mensagem" />

</LinearLayout>
```

Listagem B.25 – Cards Order Items Layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/order_item_root"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="4dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginTop="4dp"
    android:orientation="vertical">

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="140dp"
        android:background="?attr/selectableItemBackground">

        <ImageView
```

```
        android:id="@+id/order_item_image"
        android:layout_width="72dp"
        android:layout_height="72dp"
        android:layout_marginBottom="16dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="16dp"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:srcCompat="@drawable/ic_image_black_vector" />


    <TextView
        android:id="@+id/order_item_code_label"
        style="@style/OrdersList.Labels"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:text="@string/order_item_code_label"
        app:layout_constraintBottom_toTopOf="@+id/order_item_quantity_label"
        app:layout_constraintLeft_toRightOf="@+id/order_item_image"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_chainStyle="spread" />

    <TextView
        android:id="@+id/order_item_quantity_label"
        style="@style/OrdersList.Labels"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/order_item_quantity_label"
        app:layout_constraintBottom_toTopOf="@+id/order_item_description_label"
        app:layout_constraintLeft_toLeftOf="@+id/order_item_code_label"
        app:layout_constraintTop_toBottomOf="@+id/order_item_code_label" />

    <TextView
        android:id="@+id/order_item_description_label"
        style="@style/OrdersList.Labels"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/order_item_description_label"
        app:layout_constraintBottom_toTopOf="@+id/order_item_description_value"
        app:layout_constraintLeft_toLeftOf="@+id/order_item_code_label"
        app:layout_constraintTop_toBottomOf="@+id/order_item_quantity_label" />


    <TextView
        android:id="@+id/order_item_code_value"
        style="@style/OrdersList.Values"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="4dp"
        app:layout_constraintLeft_toRightOf="@+id/order_item_code_label"
        app:layout_constraintTop_toTopOf="@+id/order_item_code_label"
        tools:text="#15447" />

    <TextView
        android:id="@+id/order_item_quantity_value"
        style="@style/OrdersList.Values"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
            android:layout_marginLeft="4dp"
            app:layout_constraintLeft_toRightOf="@+id/order_item_quantity_label"
            app:layout_constraintTop_toTopOf="@+id/order_item_quantity_label"
            tools:text="3" />

        <TextView
            android:id="@+id/order_item_description_value"
            style="@style/OrdersList.Values"
            android:layout_width="0dp"
            android:layout_height="wrap_content"
            android:layout_marginBottom="8dp"
            android:ellipsize="end"
            android:maxLines="3"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toLeftOf="@+id/order_item_description_label"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/order_item_description_label"
            tools:text="Cerveja Unika West Coast" />

    </android.support.constraint.ConstraintLayout>
</android.support.v7.widget.CardView>
```

Listagem B.26 – Cards Orders List Layout

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/order_root"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginBottom="4dp"
    android:layout_marginLeft="8dp"
    android:layout_marginRight="8dp"
    android:layout_marginTop="4dp"
    android:orientation="vertical">

    <android.support.constraint.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/selectableItemBackground">

        <TextView
            android:id="@+id/order_id_label"
            style="@style/OrdersList.Labels"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="16dp"
            android:layout_marginTop="8dp"
            android:text="@string/orders_list_id_label"
            app:layout_constraintBottom_toTopOf="@+id/order_ship_type_label"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:layout_constraintVertical_chainStyle="packed" />

        <TextView
            android:id="@+id/order_ship_type_label"
            style="@style/OrdersList.Labels"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
```

```
        android:layout_marginTop="4dp"
        android:text="@string/orders_list_ship_type_label"
        app:layout_constraintBottom_toTopOf="@+id/order_size_label"
        app:layout_constraintLeft_toLeftOf="@+id/order_id_label"
        app:layout_constraintTop_toBottomOf="@+id/order_id_label" />

    <TextView
        android:id="@+id/order_size_label"
        style="@style/OrdersList.Labels"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        android:text="@string/orders_list_size_label"
        app:layout_constraintBottom_toTopOf="@+id/order_processed_at_label"
        app:layout_constraintLeft_toLeftOf="@+id/order_id_label"
        app:layout_constraintTop_toBottomOf="@+id/order_ship_type_label" />

    <TextView
        android:id="@+id/order_processed_at_label"
        style="@style/OrdersList.Labels"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="4dp"
        android:text="@string/orders_list_processed_at_label"
        app:layout_constraintBottom_toTopOf="@+id/order_last_modification_date_label"
        app:layout_constraintLeft_toLeftOf="@+id/order_id_label"
        app:layout_constraintTop_toBottomOf="@+id/order_size_label" />

    <TextView
        android:id="@+id/order_last_modification_date_label"
        style="@style/OrdersList.Labels"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="8dp"
        android:layout_marginLeft="0dp"
        android:layout_marginTop="4dp"
        android:text="@string/orders_list_last_modification_label"
        app:layout_constraintBottom_toTopOf="@+id/order_list_separator"
        app:layout_constraintLeft_toLeftOf="@+id/order_id_label"
        app:layout_constraintTop_toBottomOf="@+id/order_processed_at_label" />

    <TextView
        android:id="@+id/order_id_value"
        style="@style/OrdersList.Values"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="4dp"
        android:layout_marginStart="4dp"
        app:layout_constraintLeft_toRightOf="@+id/order_id_label"
        app:layout_constraintTop_toTopOf="@+id/order_id_label"
        tools:text="#15447" />

    <TextView
        android:id="@+id/order_ship_type_value"
        style="@style/OrdersList.Values"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="4dp"
        android:layout_marginStart="4dp"
        app:layout_constraintLeft_toRightOf="@+id/order_ship_type_label"
```

```
        app:layout_constraintTop_toTopOf="@+id/order_ship_type_label"
        tools:text="Sedex" />

    <TextView
        android:id="@+id/order_size_value"
        style="@style/OrdersList.Values"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="4dp"
        android:layout_marginStart="4dp"
        app:layout_constraintLeft_toRightOf="@+id/order_size_label"
        app:layout_constraintTop_toTopOf="@+id/order_size_label"
        tools:text="5" />

    <TextView
        android:id="@+id/order_processed_at_value"
        style="@style/OrdersList.Values"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="4dp"
        android:ellipsize="end"
        android:maxLines="1"
        app:layout_constraintLeft_toRightOf="@+id/order_processed_at_label"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="@+id/order_processed_at_label"
        tools:text="09/07/2017␣ s ␣22:00" />

    <TextView
        android:id="@+id/order_last_modification_date_value"
        style="@style/OrdersList.Values"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginEnd="8dp"
        android:layout_marginLeft="4dp"
        android:layout_marginRight="8dp"
        android:layout_marginStart="4dp"
        android:ellipsize="end"
        android:maxLines="1"
        app:layout_constraintLeft_toRightOf="@+id/order_last_modification_date_label"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="@+id/order_last_modification_date_label"
        tools:text="09/07/2017␣ s ␣22:00" />

    <View
        android:id="@+id/order_list_separator"
        android:layout_width="0dp"
        android:layout_height="1dp"
        android:layout_marginBottom="48dp"
        android:layout_marginTop="16dp"
        android:background="@color/light_grey"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/order_last_modification_date_value" />

    <TextView
        android:id="@+id/order_action_text"
```

```
            android:layout_width="wrap_content"
            android:layout_height="48dp"
            android:gravity="center"
            android:textAllCaps="true"
            android:textColor="@color/primary_text"
            android:textSize="@dimen/regular_text_size"
            android:textStyle="bold"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintHorizontal_chainStyle="packed"
            app:layout_constraintLeft_toLeftOf="parent"
            app:layout_constraintRight_toLeftOf="@+id/order_action_icon"
            app:layout_constraintTop_toBottomOf="@+id/order_list_separator"
            tools:text="@string/orders_list_action_process" />

        <ImageView
            android:id="@+id/order_action_icon"
            android:layout_width="24dp"
            android:layout_height="24dp"
            android:layout_marginLeft="4dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintLeft_toRightOf="@+id/order_action_text"
            app:layout_constraintRight_toRightOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/order_list_separator"
            app:srcCompat="@drawable/ic_right_black_vector" />

    </android.support.constraint.ConstraintLayout>

</android.support.v7.widget.CardView>
```

Listagem B.27 – View Value-Label

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.constraint.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/androi
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:background="?selectableItemBackground">

    <ImageView
        android:id="@+id/value_label_view_icon"
        android:layout_width="40dp"
        android:layout_height="40dp"
        android:visibility="gone"
        app:layout_constraintBottom_toTopOf="@+id/value_label_view_value"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_chainStyle="packed" />

    <TextView
        android:id="@+id/value_label_view_value"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textSize="@dimen/large_title_text_size"
        android:textStyle="bold"
        android:maxLines="1"
        android:paddingLeft="4dp"
        android:paddingRight="4dp"
```

```
        app:layout_constraintBottom_toTopOf="@+id/value_label_view_label"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/value_label_view_icon"
        tools:text="Value" />

    <TextView
        android:id="@+id/value_label_view_label"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginLeft="16dp"
        android:layout_marginRight="16dp"
        android:layout_marginTop="8dp"
        android:ellipsize="end"
        android:gravity="center"
        android:maxLines="2"
        android:textSize="@dimen/large_text_size"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/value_label_view_value"
        tools:text="Label" />

</android.support.constraint.ConstraintLayout>
```

B.1.2 Entity

Listagem B.28 – CodesToProcess

```
package br.com.aaascp.gerenciadordepedidos.entity;


import android.os.Parcelable;

import com.google.auto.value.AutoValue;

import java.util.Map;

import br.com.aaascp.gerenciadordepedidos.util.MathUtils;

/**
 * Created by andre on 25/09/17.
 */

@AutoValue
public abstract class CodesToProcess implements Parcelable {

    public enum Status {
        SUCCESS,
        CODE_ALREADY_PROCESSED,
        CODE_INVALID
    }

    public abstract Map<String, Integer> codes();

    public abstract int orderId();

    public Status process(String code) {
        if(!codes().containsKey(code)) {
```

```
            return Status.CODE_INVALID;
        }

        int leftItems = codes().get(code);

        if (leftItems > 0) {
            codes().put(code, leftItems - 1);
            return Status.SUCCESS;
        }

        return Status.CODE_ALREADY_PROCESSED;
    }

    public int itemsLeft() {
        return MathUtils.reduce(codes().values());
    }


    public static CodesToProcess create(Map<String, Integer> codes, int orderId) {
        return new AutoValue_CodesToProcess(codes, orderId);
    }
}
```

Listagem B.29 – CustomerInfo

```
package br.com.aaascp.gerenciadordepedidos.entity;

import android.os.Parcelable;

import com.google.auto.value.AutoValue;

/**
 * Created by andre on 20/09/17.
 */

@AutoValue
public abstract class CustomerInfo implements Parcelable {

    public abstract int id();

    public abstract String name();

    public static Builder builder() {
        return new AutoValue_CustomerInfo.Builder();
    }

    @AutoValue.Builder
    public abstract static class Builder {
        public abstract Builder id(int value);

        public abstract Builder name(String value);

        public abstract CustomerInfo build();
    }
}
```

Listagem B.30 – NullOrderFilterList

```
package br.com.aaascp.gerenciadordepedidos.entity;
```

```
import android.os.Parcelable;

import com.google.auto.value.AutoValue;

import java.util.ArrayList;
import java.util.List;

import br.com.aaascp.gerenciadordepedidos.repository.filter.OrderFilter;

/**
 * Created by andre on 27/09/17.
 */

@AutoValue
public abstract class NullOrderFilterList implements Parcelable {

    public abstract List<OrderFilter> filters();

    public static OrderFilterList create() {
        List<OrderFilter> filters = new ArrayList<>(0);
        return new AutoValue_OrderFilterList(filters);
    }
}
```

Listagem B.31 – Order

```
package br.com.aaascp.gerenciadordepedidos.entity;

import android.os.Parcelable;
import android.support.annotation.Nullable;

import com.google.auto.value.AutoValue;

import java.util.HashMap;
import java.util.Map;

import br.com.aaascp.gerenciadordepedidos.util.StringUtils;

/**
 * Created by andre on 10/07/17.
 */
@AutoValue
public abstract class Order implements Parcelable {

    public static final int INVALID_ORDER_ID = -1;

    public abstract int id();

    public abstract ShipmentInfo shipmentInfo();

    public abstract CustomerInfo customerInfo();

    public abstract Map<String, OrderItem> items();

    public abstract int size();

    @Nullable
    public abstract String processedAt();
```

```
    public abstract String lastModifiedAt();

    public CodesToProcess codesToProcess() {
        Map<String, Integer> codes = new HashMap<>(items().size());

        for (String code : items().keySet()) {
            codes.put(
                    code,
                    isProcessed() ? 0 : items().get(code).quantity());
        }

        return CodesToProcess.create(codes, id());
    }

    public boolean isProcessed() {
        return !StringUtils.isNullOrEmpty(processedAt());
    }

    public abstract Order withProcessedAt(String processedAt);

    public static Builder builder() {
        return new AutoValue_Order.Builder();
    }

    @AutoValue.Builder
    public abstract static class Builder {
        public abstract Builder id(int value);

        public abstract Builder shipmentInfo(ShipmentInfo value);

        public abstract Builder customerInfo(CustomerInfo value);

        public abstract Builder items(Map<String, OrderItem> value);

        public abstract Builder processedAt(String value);

        public abstract Builder lastModifiedAt(String value);

        public abstract Builder size(int value);

        public abstract Order build();
    }
}
```

Listagem B.32 – OrderFilterList

```
package br.com.aaascp.gerenciadordepedidos.entity;

import android.os.Parcelable;

import com.google.auto.value.AutoValue;

import java.util.List;

import br.com.aaascp.gerenciadordepedidos.repository.filter.OrderFilter;

/**
 * Created by andre on 27/09/17.
 */
```

```
@AutoValue
public abstract class OrderFilterList implements Parcelable{

    public abstract List<OrderFilter> filters();

    public static OrderFilterList create(List<OrderFilter> filters) {
        return new AutoValue_OrderFilterList(filters);
    }
}
```

Listagem B.33 – OrderItem

```
package br.com.aaascp.gerenciadordepedidos.entity;

import android.os.Parcelable;
import android.support.annotation.Nullable;

import com.google.auto.value.AutoValue;

/**
 * Created by andre on 20/09/17.
 */

@AutoValue
public abstract class OrderItem implements Parcelable {

    public abstract int id();

    public abstract String code();

    public abstract String description();

    @Nullable
    public abstract String imageUrl();

    public abstract int quantity();

    public static Builder builder() {
        return new AutoValue_OrderItem.Builder();
    }

    @AutoValue.Builder
    public abstract static class Builder {
        public abstract Builder id(int value);

        public abstract Builder code(String value);

        public abstract Builder description(String value);

        public abstract Builder imageUrl(String value);

        public abstract Builder quantity(int value);

        public abstract OrderItem build();
    }
}
```

Listagem B.34 – ShipmentInfo

```
package br.com.aaascp.gerenciadordepedidos.entity;
```

```
import android.os.Parcelable;

import com.google.auto.value.AutoValue;

/**
 * Created by andre on 20/09/17.
 */

@AutoValue
public abstract class ShipmentInfo implements Parcelable {

    public abstract String shipType();

    public abstract String address();

    public static Builder builder() {
        return new $AutoValue_ShipmentInfo.Builder();
    }

    @AutoValue.Builder
    public abstract static class Builder {
        public abstract Builder shipType(String value);

        public abstract Builder address(String value);

        public abstract ShipmentInfo build();
    }
}
```

B.1.3 Repository

Listagem B.35 – CommomOrdersRepository

```
package br.com.aaascp.gerenciadordepedidos.repository;

import java.util.ArrayList;
import java.util.List;

import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.repository.callback.DataSourceCallback;
import br.com.aaascp.gerenciadordepedidos.repository.callback.RepositoryCallback;
import br.com.aaascp.gerenciadordepedidos.repository.dao.OrdersDataSource;
import br.com.aaascp.gerenciadordepedidos.repository.filter.OrderFilter;

/**
 * Created by andre on 18/09/17.
 */

public class CommonOrdersRepository implements OrdersRepository {

    private OrdersDataSource ordersDataSource;

    public CommonOrdersRepository(OrdersDataSource ordersDataSource) {
        this.ordersDataSource = ordersDataSource;
    }

    @Override
```

```
    public void getOrder(
            int id,
            final RepositoryCallback<Order> callback) {

        ordersDataSource.load(
                id,
                new DataSourceCallback<Order>() {
                    @Override
                    public void onSuccess(Order data) {
                        callback.onSuccess(data);
                    }

                    @Override
                    public void onError(List<String> errors) {
                        callback.onError(errors);
                    }
                });
    }

    @Override
    public void getList(
            OrderFilter filter,
            RepositoryCallback<List<Order>> callback) {

        List<OrderFilter> filterList = new ArrayList<>(1);
        filterList.add(filter);

        getList(OrderFilterList.create(filterList), callback);
    }

    @Override
    public void getList(
            OrderFilterList filterList,
            final RepositoryCallback<List<Order>> callback) {

        ordersDataSource.load(
                filterList,
                new DataSourceCallback<List<Order>>() {
                    @Override
                    public void onSuccess(List<Order> data) {
                        callback.onSuccess(data);
                    }

                    @Override
                    public void onError(List<String> errors) {
                        callback.onError(errors);
                    }
                });
    }

    @Override
    public void save(Order order) {
        ordersDataSource.save(order);
    }
}
```

Listagem B.36 – DataSourceCallback

```
package br.com.aaascp.gerenciadordepedidos.repository.callback;
```

```
import java.util.List;

/**
 * Created by andre on 03/10/17.
 */

public abstract class DataSourceCallback<T> {

    public void onSuccess(T data) {
    }

    public void onError(List<String> errors) {
    }
}
```

Listagem B.37 – IdFilter

```
package br.com.aaascp.gerenciadordepedidos.repository.filter;

import android.os.Parcelable;

import com.google.auto.value.AutoValue;

import java.util.HashSet;

import br.com.aaascp.gerenciadordepedidos.entity.Order;

/**
 * Created by andre on 27/09/17.
 */

@AutoValue
public abstract class IdFilter implements OrderFilter, Parcelable {

    public abstract HashSet<Integer> ids();

    @Override
    public boolean accept(OrderVisitor visitor, Order order) {
        return visitor.filter(this, order);
    }

    public static IdFilter create(HashSet<Integer> ids) {
        return new AutoValue_IdFilter(ids);
    }
}
```

Listagem B.38 – OrderFilter

```
package br.com.aaascp.gerenciadordepedidos.repository.filter;

import br.com.aaascp.gerenciadordepedidos.entity.Order;

/**
 * Created by andre on 27/09/17.
 */

public interface OrderFilter {

     boolean accept(OrderVisitor visitor, Order order);
}
```

Listagem B.39 – OrdersDataSource

```
package br.com.aaascp.gerenciadordepedidos.repository.dao;

import java.util.List;

import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.repository.callback.DataSourceCallback;
import br.com.aaascp.gerenciadordepedidos.repository.filter.OrderVisitor;

/**
 * Created by andre on 27/09/17.
 */

public interface OrdersDataSource extends OrderVisitor {

    void save(Order order);

    void load(OrderFilterList filterList, DataSourceCallback<List<Order>> callback);

    void load(int id, DataSourceCallback<Order> callback);
}
```

Listagem B.40 – OrdersFakeDataSource

```
package br.com.aaascp.gerenciadordepedidos.repository.dao;

import android.util.ArrayMap;

import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
import java.util.Map;

import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.factory.OrdersFactory;
import br.com.aaascp.gerenciadordepedidos.repository.callback.DataSourceCallback;
import br.com.aaascp.gerenciadordepedidos.repository.filter.IdFilter;
import br.com.aaascp.gerenciadordepedidos.repository.filter.StatusFilter;

/**
 * Created by andre on 03/10/17.
 */

public class OrdersFakeDataSource implements OrdersDataSource {

    public static final String ERROR_MESSAGE = "ERROR";
    private static final int ORDER_ERROR_ID = -1000;
    private static final int ORDER_COMMUNICATION_ERROR_ID = -2000;

    private static Map<Integer, Order> ORDERS_DATA = new ArrayMap<>();

    @Override
    public void save(Order order) {
        ORDERS_DATA.put(order.id(), order);
    }

    @Override
```

```
public void load(
        OrderFilterList filterList,
        final DataSourceCallback<List<Order>> callback) {

    for (Order order : ORDERS_DATA.values()) {
        if (order.id() == ORDER_ERROR_ID) {
            callback.onError(Collections.singletonList(ERROR_MESSAGE));
            return;
        } else if (order.id() == ORDER_COMMUNICATION_ERROR_ID) {
            callback.onError(null);
            return;
        }
    }

    List<Order> orders = new ArrayList<>(ORDERS_DATA.size());
    orders.addAll(ORDERS_DATA.values());
    callback.onSuccess(orders);

}

@Override
public void load(int id, final DataSourceCallback<Order> callback) {
    final Order order = ORDERS_DATA.get(id);

    if (order.id() == ORDER_ERROR_ID) {
        callback.onError(Collections.singletonList(ERROR_MESSAGE));
        return;
    } else if (order.id() == ORDER_COMMUNICATION_ERROR_ID) {
        callback.onError(null);
        return;
    }

    callback.onSuccess(copyOrder(order));
}

public static Order load(int id) {
    return ORDERS_DATA.get(id);
}

@Override
public boolean filter(IdFilter filter, Order order) {
    return true;
}

@Override
public boolean filter(StatusFilter filter, Order order) {
    return true;
}

private Order copyOrder(Order order) {
    return Order.builder()
            .id(order.id())
            .items(order.items())
            .processedAt(order.processedAt())
            .shipmentInfo(order.shipmentInfo())
            .customerInfo(order.customerInfo())
            .lastModifiedAt(order.lastModifiedAt())
            .size(order.size())
            .build();
}
```

```
    public static void clear() {
        ORDERS_DATA.clear();
    }

    public static void addOrder(Order order) {
        ORDERS_DATA.put(order.id(), order);
    }

    public static Order createOrderError() {
        return OrdersFactory.createOrder(ORDER_ERROR_ID, false);
    }

    public static Order createOrderCommunicationError() {
        return OrdersFactory.createOrder(ORDER_COMMUNICATION_ERROR_ID, false);
    }
}
```

Listagem B.41 – OrdersMemoryDataSource

```
package br.com.aaascp.gerenciadordepedidos.repository.dao;

import android.os.Handler;
import android.util.SparseArray;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import br.com.aaascp.gerenciadordepedidos.entity.CustomerInfo;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.entity.OrderItem;
import br.com.aaascp.gerenciadordepedidos.entity.ShipmentInfo;
import br.com.aaascp.gerenciadordepedidos.repository.callback.DataSourceCallback;
import br.com.aaascp.gerenciadordepedidos.repository.filter.OrderFilter;
import br.com.aaascp.gerenciadordepedidos.repository.filter.IdFilter;
import br.com.aaascp.gerenciadordepedidos.repository.filter.StatusFilter;
import br.com.aaascp.gerenciadordepedidos.util.DateFormatterUtils;

/**
 * Created by andre on 22/09/17.
 */

public final class OrdersMemoryDataSource implements OrdersDataSource {

    private static final int NETWORK_DELAY = 500;
    private static SparseArray<Order> ORDERS_DATA = new SparseArray<>();

    static {
        ORDERS_DATA = new SparseArray<>(4);
        addOrder(1000, false);
        addOrder(1001, false);
        addOrder(1002, true);
        addOrder(1003, false);
    }

    @Override
    public void save(Order order) {
```

```
        ORDERS_DATA.put(order.id(), order);
    }

    @Override
    public void load(
            final OrderFilterList filterList,
            final DataSourceCallback<List<Order>> callback) {

        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {

                List<OrderFilter> filters = filterList.filters();
                final List<Order> filteredOrders = new ArrayList<>();

                for (int i = 0; i < ORDERS_DATA.size(); i++) {
                    Order order = ORDERS_DATA.valueAt(i);
                    boolean isFiltered;

                    for (OrderFilter filter : filters) {
                        isFiltered = filter.accept(OrdersMemoryDataSource.this, order);

                        if (isFiltered) {
                            filteredOrders.add(copyOrder(order));
                            break;
                        }
                    }
                }

                callback.onSuccess(filteredOrders);
            }
        }, NETWORK_DELAY);
    }

    @Override
    public void load(final int id, final DataSourceCallback<Order> callback) {


        Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            @Override
            public void run() {
                Order order = ORDERS_DATA.get(id);
                callback.onSuccess(copyOrder(order));
            }
        }, NETWORK_DELAY);
    }

    @Override
    public boolean filter(IdFilter filter, Order order) {
        return filter.ids().contains(order.id());
    }

    @Override
    public boolean filter(StatusFilter filter, Order order) {
        boolean isProcessed = order.isProcessed();

        switch (filter.status()) {
            case PROCESSED:
```

```
                return isProcessed;
            case TO_PROCESS:
                return !isProcessed;
            case ALL:
                return true;
            default:
                throw new IllegalArgumentException();
        }
    }

    private Order copyOrder(Order order) {
        return Order.builder()
                .id(order.id())
                .items(order.items())
                .processedAt(order.processedAt())
                .shipmentInfo(order.shipmentInfo())
                .customerInfo(order.customerInfo())
                .lastModifiedAt(order.lastModifiedAt())
                .size(order.size())
                .build();
    }

    private static void addOrder(int id, boolean processed) {
        ORDERS_DATA.append(id, orderFactory(id, processed));
    }

    private static Order orderFactory(int id, boolean processed) {
        Map<String, OrderItem> items = new HashMap<>();

        OrderItem item1 = OrderItem.builder()
                .id(id * 10 + 1)
                .code("314159265359")
                .description("Cerveja 1")
                .imageUrl("")
                .quantity(2)
                .build();

        items.put("314159265359", item1);

        ShipmentInfo shipmentInfo = ShipmentInfo.builder()
                .shipType("Sedex")
                .address("Avenida Engenheiro Max de Souza, 1293, Coqueiros")
                .build();

        CustomerInfo customerInfo = CustomerInfo.builder()
                .id(1000)
                .name("Andr  Alex Araujo")
                .build();

        return Order.builder()
                .id(id)
                .shipmentInfo(shipmentInfo)
                .customerInfo(customerInfo)
                .items(items)
                .size(2)
                .processedAt(processed ? DateFormatterUtils.getDateHourInstance().now() : null)
                .lastModifiedAt("09/07/2017 s 22:00")
                .build();
    }
}
```

Listagem B.42 – OrdersRepository

```
package br.com.aaascp.gerenciadordepedidos.repository;

import java.util.List;

import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.repository.callback.RepositoryCallback;
import br.com.aaascp.gerenciadordepedidos.repository.filter.OrderFilter;

/**
 * Created by andre on 18/09/17.
 */

public interface OrdersRepository {

    void getOrder(
            int id,
            final RepositoryCallback<Order> callback);

    void getList(
            OrderFilter filter,
            RepositoryCallback<List<Order>> callback);

    void getList(
            OrderFilterList filterList,
            final RepositoryCallback<List<Order>> callback);

    void save(Order order);
}
```

Listagem B.43 – OrderVisitor

```
package br.com.aaascp.gerenciadordepedidos.repository.filter;

import br.com.aaascp.gerenciadordepedidos.entity.Order;

/**
 * Created by andre on 27/09/17.
 */

public interface OrderVisitor {

    boolean filter(IdFilter filter, Order order);

    boolean filter(StatusFilter filter, Order order);
}
```

Listagem B.44 – RepositoryCallback

```
package br.com.aaascp.gerenciadordepedidos.repository.callback;

import java.util.List;

/**
 * Created by andre on 10/07/17.
 */

public abstract class RepositoryCallback<T> {
```

```
    public void onSuccess(T data) {
    }

    public void onError(List<String> errors) {
    }
}
```

Listagem B.45 – StatusFilter

```
package br.com.aaascp.gerenciadordepedidos.repository.filter;


import android.os.Parcelable;

import com.google.auto.value.AutoValue;

import br.com.aaascp.gerenciadordepedidos.entity.Order;

/**
 * Created by andre on 27/09/17.
 */

@AutoValue
public abstract class StatusFilter implements OrderFilter, Parcelable {

    public enum Status {
        PROCESSED,
        TO_PROCESS,
        ALL
    }

    public abstract Status status();

    @Override
    public boolean accept(OrderVisitor visitor, Order order) {
        return visitor.filter(this, order);
    }

    public static StatusFilter create(Status status) {
        return new AutoValue_StatusFilter(status);
    }
}
```

B.1.4 Utils

Listagem B.46 – DateFormatterUtils

```
package br.com.aaascp.gerenciadordepedidos.util;

import java.text.SimpleDateFormat;
import java.util.Date;

/**
 * Created by andre on 27/09/17.
 */

public final class DateFormatterUtils {
```

```
    private static final String FORMAT_DATE_HOUR = "dd/MM/yyyy ' s ' HH:mm";

    private final SimpleDateFormat formatter;

    private DateFormatterUtils(String format) {
        this.formatter = new SimpleDateFormat(format);
    }

    public static DateFormatterUtils getDateHourInstance() {
        return new DateFormatterUtils(FORMAT_DATE_HOUR);
    }

    public String format(Date date) {
        return formatter.format(date);
    }

    public String now() {
        return format(new Date());
    }

}
```

Listagem B.47 – MathUtils

```
package br.com.aaascp.gerenciadordepedidos.util;

import java.util.Collection;

/**
 * Created by andre on 23/09/17.
 */

public final class MathUtils {

    public static int reduce(Collection<Integer> collection) {
        int acc = 0;

        for(int item : collection) {
            acc += item;
        }

        return  acc;
    }
}
```

Listagem B.48 – PermissionUtils

```
package br.com.aaascp.gerenciadordepedidos.util;

import android.Manifest;
import android.content.Context;
import android.content.pm.PackageManager;
import android.support.v4.app.ActivityCompat;
import android.support.v4.content.ContextCompat;

import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.DialogUtils;
```

```
/**
 * Created by andre on 28/09/17.
 */

public final class PermissionUtils {

    public static void requestPermissionForCamera(BaseActivity activity, int requestCode) {
        requestPermissionRationaleForCamera(activity, requestCode);
    }

    public static boolean isCameraEnabled(BaseActivity activity) {
        return isEnabled(activity, Manifest.permission.CAMERA);
    }

    private static void requestPermissionRationaleForCamera(BaseActivity activity, int requestCode) {
        requestPermissionRational(
                activity,
                Manifest.permission.CAMERA,
                activity.getString(R.string.permission_name_camera),
                requestCode);
    }

    private static void requestPermissionRational(
            BaseActivity activity,
            String permission,
            String permissionName,
            int requestCode) {

        if (!isEnabled(activity, permission)) {
            if (shouldShowRequestPermissionRationale(activity, permission)) {
                DialogUtils.showPermissionError(activity, permissionName);
            } else {
                requestPermission(activity, permission, requestCode);
            }
        }
    }

    private static boolean shouldShowRequestPermissionRationale(
            BaseActivity activity,
            String permission) {

        return ActivityCompat.shouldShowRequestPermissionRationale(
                activity,
                permission);
    }

    private static void requestPermission(
            BaseActivity activity,
            String permission,
            int requestCode) {

        ActivityCompat.requestPermissions(
                activity,
                new String[]{permission},
                requestCode);
    }

    private static boolean isEnabled(Context context, String permission) {
        return isPermissionEnabled(
                ContextCompat.checkSelfPermission(
```

```
                        context,
                        permission));
    }

    private static boolean isPermissionEnabled(int permissionCheck) {
        return permissionCheck == PackageManager.PERMISSION_GRANTED;
    }


}
```

Listagem B.49 – StringUtils

```
package br.com.aaascp.gerenciadordepedidos.util;

/**
 * Created by andre on 22/09/17.
 */

public final class StringUtils {

    public static boolean isNullOrEmpty(String string) {
        return string == null ||
                string.equals("");
    }
}
```

Listagem B.50 – DialogUtils

```
package br.com.aaascp.gerenciadordepedidos.presentation.util;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.support.annotation.StringRes;
import android.text.InputType;
import android.text.method.DigitsKeyListener;
import android.widget.EditText;

import java.util.HashSet;

import br.com.aaascp.gerenciadordepedidos.R;

/**
 * Created by andre on 28/09/17.
 */

public class DialogUtils {

    private static AlertDialog.Builder getGenericBuilder(
            Context context,
            @StringRes int title,
            @StringRes int message) {

        final AlertDialog.Builder builder = new AlertDialog.Builder(context);

        builder.setTitle(title)
                .setMessage(message)
                .setNegativeButton(
```

```
                    R.string.dialog_cancel,
                    new android.app.AlertDialog.OnClickListener() {
                        @Override
                        public void onClick(DialogInterface dialog, int id) {
                            dialog.dismiss();
                        }
                    });

        return builder;
    }

    public static void showGenericDialog(
            Context context,
            @StringRes int title,
            @StringRes int message,
            DialogInterface.OnClickListener positiveListener) {

        getGenericBuilder(
                context,
                title,
                message)
                .setPositiveButton(
                        R.string.dialog_ok,
                        positiveListener)
                .show();
    }

    public static void showGetIntValues(
            Context context,
            @StringRes int title,
            @StringRes int message,
            final IntValuesListener listener) {

        final EditText editText = new EditText(context);
        editText.setInputType(InputType.TYPE_CLASS_NUMBER);
        editText.setKeyListener(DigitsKeyListener.getInstance("0123456789,"));

        getGenericBuilder(
                context,
                title,
                message)
                .setView(editText)
                .setPositiveButton(
                        R.string.dialog_ok,
                        new DialogInterface.OnClickListener() {

                            @Override
                            public void onClick(DialogInterface dialog, int id) {
                                final String[] valuesText = editText.getText()
                                        .toString()
                                        .split(",");

                                HashSet<Integer> values = new HashSet<>();
                                boolean error = false;

                                for (String value : valuesText) {
                                    try {
                                        int intValue = Integer.valueOf(value);
                                        values.add(intValue);
                                    } catch (NumberFormatException exception) {
```

```
                                        listener.onError();
                                        error = true;
                                        break;
                                    }
                                }

                                if (!error) {
                                    listener.onValues(values);
                                }
                            }
                        })
                .show();
    }

    public static void showError(
            Context context,
            String title,
            String message) {

        AlertDialog.Builder builder = new AlertDialog.Builder(context);

        builder.setTitle(title)
                .setMessage(message);

        builder.setPositiveButton(
                R.string.dialog_ok,
                new DismissListener());

        builder.create().show();
    }

    public static void showPermissionError(Context context, String permission) {
        showError(
                context,
                context.getString(R.string.error_permission_title),
                String.format(
                        context.getString(R.string.error_permission_message),
                        permission));
    }

    public interface IntValuesListener {
        void onValues(HashSet<Integer> values);

        void onError();
    }

    private static final class DismissListener implements DialogInterface.OnClickListener {

        @Override
        public void onClick(DialogInterface dialog, int id) {
            dialog.dismiss();
        }
    }
}
```

Listagem B.51 – SnackBarUtils

```
package br.com.aaascp.gerenciadordepedidos.presentation.util;

import android.support.design.widget.Snackbar;
```

```
import android.view.View;
import android.widget.TextView;

/**
 * Created by andre on 28/09/17.
 */

public final class SnackBarUtils {

    public static void showWithCenteredText(View parent, String message) {

        Snackbar snackBar = Snackbar.make(
                parent,
                message,
                Snackbar.LENGTH_LONG);

        View view = snackBar.getView();

        TextView snackBarText = (TextView) view.findViewById(android.support.design.R.id.snackbar_text);
        snackBarText.setTextAlignment(View.TEXT_ALIGNMENT_CENTER);

        snackBar.show();
    }
}
```

Listagem B.52 – ImageLoader

```
package br.com.aaascp.gerenciadordepedidos.presentation.util;

import android.content.Context;
import android.widget.ImageView;

import com.squareup.picasso.Picasso;

/**
 * Created by andre on 27/09/17.
 */

public class ImageLoader {

    public static void loadImage(
            Context context,
            String url,
            ImageView imageView) {

        Picasso.with(context)
                .load(url)
                .into(imageView);
    }
}
```

B.2 Código Específico versão MVC

Listagem B.53 – BarcodeProcessor

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;

import android.util.SparseArray;
```

```
import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.barcode.Barcode;

import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;

/**
 * Created by andre on 22/09/17.
 */

final class BarcodeProcessor implements Detector.Processor<Barcode> {

    private final BaseActivity baseActivity;
    private final OnItemProcessedListener listener;

    BarcodeProcessor(
            BaseActivity baseActivity,
            OnItemProcessedListener listener) {

        this.baseActivity = baseActivity;
        this.listener = listener;
    }

    @Override
    public void receiveDetections(Detector.Detections<Barcode> detections) {

        final SparseArray<Barcode> barcodes = detections.getDetectedItems();

        if (barcodes.size() != 0) {
            final String code = barcodes.valueAt(0).displayValue;

            baseActivity.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    listener.onItemProcessed(code);
                }
            });
        }
    }

    @Override
    public void release() {

    }
}
```

Listagem B.54 – BarcodeProcessorActivity

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.ColorRes;
import android.support.annotation.Nullable;
import android.support.v4.content.ContextCompat;
import android.support.v7.widget.Toolbar;
import android.view.SurfaceView;
```

```
import android.view.View;
import android.widget.TextView;

import com.google.android.gms.vision.CameraSource;
import com.google.android.gms.vision.barcode.BarcodeDetector;

import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.SnackBarUtils;
import butterknife.BindView;
import butterknife.ButterKnife;

import static android.content.res.Configuration.ORIENTATION_LANDSCAPE;
import static android.content.res.Configuration.ORIENTATION_PORTRAIT;

/**
 * Created by andre on 21/09/17.
 */

public final class BarcodeProcessorActivity extends BaseActivity
        implements OnItemProcessedListener {

    public static final String EXTRA_RESULT = "EXTRA_RESULT";

    public static final String EXTRA_ORDER_ID = "EXTRA_ORDER_ID";
    public static final String EXTRA_CODES_TO_PROCESS = "EXTRA_CODES_TO_PROCESS";

    private static final int PROCESSING_TIME = 3 * 1000;

    private BarcodeDetector barcodeDetector;

    @BindView(R.id.barcode_processor_toolbar)
    Toolbar toolbar;

    @BindView(R.id.barcode_processor_preview)
    SurfaceView previewLayout;

    @BindView(R.id.barcode_processor_root)
    View root;

    @BindView(R.id.barcode_processor_guide)
    View guide;

    @BindView(R.id.barcode_processor_title)
    TextView title;

    @BindView(R.id.barcode_processor_items_left)
    TextView itemsLeft;

    private CodesToProcess codesToProcess;
    private int orderId;
    private boolean ready;

    public static Intent getIntentForOrder(
            Context context,
            int orderId,
            CodesToProcess codesToProcess) {

        Intent intent = new Intent(
```

```
                context,
                BarcodeProcessorActivity.class);

        intent.putExtra(EXTRA_CODES_TO_PROCESS, codesToProcess);
        intent.putExtra(EXTRA_ORDER_ID, orderId);

        return intent;
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_barcode_processor);

        ButterKnife.bind(this);

        extractExtras();

        ready = true;

        setupToolbar();
        setItemsLeft();
        setupBarcodeDetector();
        setupCamera();
    }

    @Override
    public void onSaveInstanceState(Bundle outState) {
        super.onSaveInstanceState(outState);

        outState.putParcelable(EXTRA_CODES_TO_PROCESS, codesToProcess);
    }

    @Override
    protected void onRestoreInstanceState(Bundle savedInstanceState) {
        super.onRestoreInstanceState(savedInstanceState);

        codesToProcess = savedInstanceState.getParcelable(EXTRA_CODES_TO_PROCESS);
        setItemsLeft();
    }

    @Override
    public void finish() {
        Intent result = new Intent();
        result.putExtra(EXTRA_RESULT, codesToProcess);
        setResult(RESULT_OK, result);

        super.finish();
    }

    @Override
    public void onItemProcessed(String code) {
        if (!ready) {
            return;
        }

        ready = false;

        CodesToProcess.Status status = codesToProcess.process(code);
```

```
        showMessage(status, code);

        if (status == CodesToProcess.Status.SUCCESS) {
            showProcessing(R.color.green);
            setItemsLeft();
            checkFinish();
        } else {
            showProcessing(R.color.red);
        }
    }

    private void extractExtras() {
        Bundle extras = getIntent().getExtras();
        if (extras != null) {
            codesToProcess = extras.getParcelable(EXTRA_CODES_TO_PROCESS);
            orderId = extras.getInt(EXTRA_ORDER_ID);
        }
    }

    private void setupToolbar() {
        title.setText(
                String.format(
                        getString(R.string.barcode_processor_title),
                        orderId));

        toolbar.setNavigationIcon(R.drawable.ic_back_white_vector);
        toolbar.setNavigationOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                onBackPressed();
            }
        });
    }

    private void setItemsLeft() {
        int itemsLeftCount = codesToProcess.itemsLeft();

        String itemsLeftText =
                getResources()
                        .getQuantityString(
                                R.plurals.barcode_processor_items_left,
                                itemsLeftCount);

        if (itemsLeftCount == 0) {
            itemsLeftText = getString(R.string.barcode_processor_items_left);
        }

        itemsLeft.setText(
                String.format(
                        itemsLeftText,
                        itemsLeftCount));
    }

    private void setupBarcodeDetector() {
        barcodeDetector =
                new BarcodeDetector.Builder(this)
                        .build();

        if (!barcodeDetector.isOperational()) {
            return;
```

```
        }

        barcodeDetector.setProcessor(new BarcodeProcessor(this, this));
    }

    private void setupCamera() {
        CameraSource cameraSource = new CameraSource
                .Builder(this, barcodeDetector)
                .setAutoFocusEnabled(true)
                .build();

        previewLayout.getHolder()
                .addCallback(
                        new CameraPreview(cameraSource, this));
    }

    private void showProcessing(final @ColorRes int color) {
        final int normalColor =
                ContextCompat.getColor(
                        this,
                        android.R.color.white);

        final int highlightColor =
                ContextCompat.getColor(
                        this,
                        color);


        guide.getBackground().setTint(highlightColor);

        root.postDelayed(new Runnable() {
            @Override
            public void run() {
                guide.getBackground().setTint(normalColor);
                ready = true;
            }
        }, PROCESSING_TIME);
    }

    private void showMessage(CodesToProcess.Status status, String code) {
        String message = "";

        switch (status) {
            case SUCCESS:
                message = getString(R.string.barcode_processor_success_message);
                break;
            case CODE_ALREADY_PROCESSED:
                message = getString(R.string.barcode_processor_already_processed_message);
                break;
            case CODE_INVALID:
                message = getString(R.string.barcode_processor_invalid_message);
                break;
        }

        SnackBarUtils.showWithCenteredText(
                root,
                String.format(
                        message,
                        code));
    }
```

```
    private void checkFinish() {
        if (codesToProcess.itemsLeft() == 0) {
            showFinishDialog();
        }
    }

    private void showFinishDialog() {
        final AlertDialog.Builder builder = new AlertDialog.Builder(this);

        builder.setTitle(R.string.barcode_processor_finish_dialog_title)
                .setMessage(R.string.barcode_processor_finish_dialog_message)
                .setPositiveButton(
                        R.string.dialog_ok,
                        new AlertDialog.OnClickListener() {
                            @Override
                            public void onClick(DialogInterface dialog, int id) {
                                finish();
                            }
                        });

        builder.show();
    }
}
```

Listagem B.55 – CameraPreview

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;

import android.util.Log;
import android.view.SurfaceHolder;

import com.google.android.gms.vision.CameraSource;

import java.io.IOException;

import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import br.com.aaascp.gerenciadordepedidos.util.PermissionUtils;

/**
 * Created by andre on 21/09/17.
 */

final class CameraPreview implements SurfaceHolder.Callback {

    private static final String TAG = CameraPreview.class.getSimpleName();

    private final CameraSource cameraSource;
    private final BaseActivity activity;

    CameraPreview(CameraSource cameraSource, BaseActivity activity) {
        this.cameraSource = cameraSource;
        this.activity = activity;
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        try {
            if(PermissionUtils.isCameraEnabled(activity)) {
                cameraSource.start(holder);
```

```
            } else {
                activity.finish();
            }
        } catch (IOException e) {
            Log.e(TAG, e.getMessage());
        }
    }

    @Override
    public void surfaceChanged(
            SurfaceHolder holder,
            int format,
            int width,
            int height) {

        //
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        cameraSource.stop();
    }
}
```

Listagem B.56 – DashboardFragment

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.main.dashboard;

import android.os.Bundle;
import android.support.annotation.Nullable;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;

import br.com.aaascp.gerenciadordepedidos.Inject;
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.presentation.custom.ValueLabelView;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseFragment;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list.OrdersListActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.DialogUtils;
import br.com.aaascp.gerenciadordepedidos.repository.OrdersRepository;
import br.com.aaascp.gerenciadordepedidos.repository.callback.RepositoryCallback;
import br.com.aaascp.gerenciadordepedidos.repository.filter.OrderFilter;
import br.com.aaascp.gerenciadordepedidos.repository.filter.IdFilter;
import br.com.aaascp.gerenciadordepedidos.repository.filter.StatusFilter;
import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

/**
 * Created by andre on 18/09/17.
 */
```

```
public final class DashboardFragment extends BaseFragment {

    @BindView(R.id.dashboard_to_process)
    ValueLabelView toProcessButton;

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle save
        super.onCreateView(inflater, container, savedInstanceState);

        View view = inflater.inflate(
                R.layout.fragment_dashboard,
                container,
                false);

        ButterKnife.bind(this, view);

        return view;
    }

    @Override
    public void onStart() {
        super.onStart();

        setupToProcessButton();
    }

    private void setupToProcessButton() {
        OrdersRepository orderRepository = Inject.provideOrdersRepository();
        StatusFilter filter = StatusFilter.create(StatusFilter.Status.TO_PROCESS);

        orderRepository.getList(
                filter,
                new RepositoryCallback<List<Order>>() {
                    @Override
                    public void onSuccess(List<Order> data) {
                        Log.d("Andre", data.toString());
                        Log.d("Andre", String.valueOf(data.size()));
                        toProcessButton.setValue(
                                String.valueOf(data.size()));
                    }

                    @Override
                    public void onError(List<String> errors) {
                        toProcessButton.setValue("?");
                    }
                });

    }

    private void navigateToOrdersList(OrderFilter filter, boolean processAll) {
        List<OrderFilter> filters = new ArrayList<>();
        filters.add(filter);

        OrdersListActivity.startForContext(
                getActivity(),
                OrderFilterList.create(filters),
                processAll);
    }
```

```
    private void getIds() {
        DialogUtils.showGetIntValues(
                getActivity(),
                R.string.dashboard_orders_find_dialog_title,
                R.string.dashboard_orders_find_dialog_message,
                new DialogUtils.IntValuesListener() {
                    @Override
                    public void onValues(HashSet<Integer> values) {
                        navigateToOrdersList(
                                IdFilter.create(values),
                                values.size() > 1);
                    }

                    @Override
                    public void onError() {
                        showErrorGettingIds();
                    }
                });
    }

    private void showErrorGettingIds() {
        DialogUtils.showError(
                getActivity(),
                getString(R.string.dashboard_orders_find_dialog_error_title),
                getString(R.string.dashboard_orders_find_dialog_erro_message));
    }

    @OnClick(R.id.dashboard_to_process)
    void toProcessButton() {
        navigateToOrdersList(
                StatusFilter.create(StatusFilter.Status.TO_PROCESS),
                true);
    }

    @OnClick(R.id.dashboard_processed)
    void processedButton() {
        navigateToOrdersList(
                StatusFilter.create(StatusFilter.Status.PROCESSED),
                false);
    }

    @OnClick(R.id.dashboard_all)
    void allButton() {
        navigateToOrdersList(
                StatusFilter.create(StatusFilter.Status.ALL),
                false);
    }

    @OnClick(R.id.dashboard_find)
    void findButton() {
        getIds();
    }
}
```

Listagem B.57 – MainActivity

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.main;

import android.os.Bundle;
```

```
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import butterknife.ButterKnife;

public final class MainActivity extends BaseActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);
    }
}
```

Listagem B.58 – OnItemProcessedListener

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;

/**
 * Created by andre on 22/09/17.
 */

interface OnItemProcessedListener {

    void onItemProcessed(String code);
}
```

Listagem B.59 – OrderInfoActivity

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.info;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.TextView;

import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by andre on 27/09/17.
 */

public final class OrderInfoActivity extends BaseActivity {

    public static final String EXTRA_ORDER = "EXTRA_ORDER";

    @BindView(R.id.order_info_toolbar)
    Toolbar toolbar;

    @BindView(R.id.info_order_size_value)
    TextView orderSize;
```

```
@BindView(R.id.info_order_processed_at_value)
TextView orderProcessedAt;

@BindView(R.id.info_order_last_modification_value)
TextView orderLastModification;

@BindView(R.id.info_shipment_type_value)
TextView shipmentType;

@BindView(R.id.info_shipment_address_value)
TextView shipmentAddress;

@BindView(R.id.info_customer_id_value)
TextView customerId;

@BindView(R.id.info_customer_name_value)
TextView customerName;

private Order order;


public static void startForOrder(Context context, Order order) {
    Intent intent = new Intent(context, OrderInfoActivity.class);

    intent.putExtra(EXTRA_ORDER, order);

    context.startActivity(intent);
}

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_order_info);
    ButterKnife.bind(this);

    extractExtras();
    bindView();
    setupTitle();
}

private void extractExtras() {
    Bundle extras = getIntent().getExtras();

    if (extras != null) {
        order = extras.getParcelable(EXTRA_ORDER);
    }
}

private void setupTitle() {
    toolbar.setTitle(
            String.format(
                    getString(R.string.info_title),
                    order.id()));
    toolbar.setNavigationIcon(R.drawable.ic_back_white_vector);

    toolbar.setNavigationOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
```

```
                onBackPressed();
            }
        });
    }

    private void bindView() {
        orderSize.setText(String.valueOf(order.size()));
        orderProcessedAt.setText(getProcessedAt());
        orderLastModification.setText(order.lastModifiedAt());

        shipmentType.setText(order.shipmentInfo().shipType());
        shipmentAddress.setText(order.shipmentInfo().address());

        customerId.setText(
                String.valueOf(
                        order.customerInfo().id()));
        customerName.setText(order.customerInfo().name());
    }

    private String getProcessedAt() {
        if (order.isProcessed()) {
            return order.processedAt();
        }

        return getString(R.string.info_order_processed_at_empty);
    }
}
```

Listagem B.60 – OrderItemActivity

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.item;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;

import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.OrderItem;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.ImageLoader;
import br.com.aaascp.gerenciadordepedidos.util.StringUtils;
import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by andre on 27/09/17.
 */

public final class OrderItemActivity extends BaseActivity {

    public static final String EXTRA_ITEM = "EXTRA_ITEM";
    @BindView(R.id.order_item_toolbar)
    Toolbar toolbar;

    @BindView(R.id.order_item_image)
```

```
ImageView imageView;

@BindView(R.id.order_item_description)
TextView descriptionView;

private OrderItem item;

public static void startForItem(Context context, OrderItem item) {
    Intent intent = new Intent(context, OrderItemActivity.class);

    intent.putExtra(EXTRA_ITEM, item);

    context.startActivity(intent);
}

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_order_item);
    ButterKnife.bind(this);

    extractExtras();
    setupToolbar();
    bindView();
}

private void extractExtras() {
    Bundle extras = getIntent().getExtras();

    if (extras != null) {
        item = extras.getParcelable(EXTRA_ITEM);
    }
}

private void setupToolbar() {
    toolbar.setNavigationIcon(R.drawable.ic_close_white_vector);
    toolbar.setNavigationOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            finish();
        }
    });

    toolbar.setTitle(
            String.format(
                    getString(R.string.order_item_title),
                    item.code()));
}

private void bindView() {
    String imageUrl = item.imageUrl();
    if (!StringUtils.isNullOrEmpty(imageUrl)) {
        ImageLoader.loadImage(
                this,
                imageUrl,
                imageView);
    }

    descriptionView.setText(item.description());
```

```
    }
}
```

Listagem B.61 – OrdersFactory

```
package br.com.aaascp.gerenciadordepedidos.factory;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import br.com.aaascp.gerenciadordepedidos.entity.CustomerInfo;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderItem;
import br.com.aaascp.gerenciadordepedidos.entity.ShipmentInfo;
import br.com.aaascp.gerenciadordepedidos.util.DateFormatterUtils;

/**
 * Created by andre on 02/10/17.
 */
public class OrdersFactory {

    public static List<Order> getOrders(int size, double processedProbability) {
        List<Order> orders = new ArrayList<>(size);

        for (int i = 0; i < size; i++) {
            double rand = Math.random();
            orders.add(createOrder(3000 + i, rand < processedProbability));
        }

        return orders;
    }

    public static Order createOrder(
            int id,
            ShipmentInfo shipmentInfo,
            CustomerInfo customerInfo,
            int size,
            String processedAt,
            String lastModifiedAt) {

        List<OrderItem> itemsCatalog = new ArrayList<>(3);

        itemsCatalog.add(
                OrderItem.builder()
                        .id(1234)
                        .code("3141592653 59")
                        .description("Cerveja 1")
                        .imageUrl("")
                        .quantity(0)
                        .build());

        itemsCatalog.add(
                OrderItem.builder()
                        .id(2345)
                        .code("5012345678900")
                        .description("Cerveja 2")
                        .imageUrl("")
                        .quantity(0)
```

```
                        .build());

        itemsCatalog.add(
                OrderItem.builder()
                        .id(3456)
                        .code("7898357410015")
                        .description("Cerveja 3")
                        .imageUrl("")
                        .quantity(0)
                        .build());

        Map<String, OrderItem> items = new HashMap<>();

        int index = 0;
        for (int i = 0; i < size; i++) {
            OrderItem item = itemsCatalog.get(index);
            OrderItem newItem =
                    OrderItem.builder()
                            .id(item.id())
                            .code(item.code())
                            .description(item.description())
                            .imageUrl(item.imageUrl())
                            .quantity(item.quantity() + 1)
                            .build();

            items.put(item.code(), newItem);
            itemsCatalog.set(index, newItem);

            index = index < itemsCatalog.size() - 1 ? index + 1 : 0;
        }

        return Order.builder()
                .id(id)
                .shipmentInfo(shipmentInfo)
                .customerInfo(customerInfo)
                .items(items)
                .size(size)
                .processedAt(processedAt)
                .lastModifiedAt(lastModifiedAt)
                .build();
    }

    public static Order createOrder(int id, boolean processed) {
        return createOrder(
                id,
                ShipmentInfo.builder().shipType("Sedex").address("Endereco").build(),
                CustomerInfo.builder().id(1).name("Customer 1").build(),
                2,
                processed ? DateFormatterUtils.getDateHourInstance().now() : null,
                DateFormatterUtils.getDateHourInstance().now());
    }
}
```

Listagem B.62 – OrdersListActivity

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
```

```
import android.support.annotation.Nullable;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.View;

import java.util.List;

import br.com.aaascp.gerenciadordepedidos.Inject;
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.details.OrderDetailsActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.EmptyStateAdapter;
import br.com.aaascp.gerenciadordepedidos.repository.OrdersRepository;
import br.com.aaascp.gerenciadordepedidos.repository.callback.RepositoryCallback;
import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

/**
 * Created by andre on 09/07/17.
 */

public final class OrdersListActivity extends BaseActivity {

    public static final int REQUEST_CODE_ORDER_PROCESS = 100;

    public static final int RESULT_CODE_OK_UNIQUE = 200;
    public static final int RESULT_CODE_OK = 300;
    public static final int RESULT_CODE_SKIP = 400;
    public static final int RESULT_CODE_CLOSE = 500;


    static final String EXTRA_ORDER_FILTERS = "EXTRA_ORDER_FILTERS";
    static final String EXTRA_PROCESS_ALL = "EXTRA_PROCESS_ALL";

    @BindView(R.id.orders_list_fab)
    FloatingActionButton fab;

    @BindView(R.id.orders_list_toolbar)
    Toolbar toolbar;

    @BindView(R.id.orders_list_recycler)
    RecyclerView recyclerView;

    private OrdersRepository ordersRepository;
    private OrderFilterList filterList;
    private List<Order> orders;
    private boolean processAll;
    private int current;

    public static void startForContext(
            Context context,
            OrderFilterList filters,
            boolean processAll) {

        Intent intent =
                new Intent(
```

```
                    context,
                    OrdersListActivity.class);

        intent.putExtra(EXTRA_ORDER_FILTERS, filters);
        intent.putExtra(EXTRA_PROCESS_ALL, processAll);

        context.startActivity(intent);
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_orders_list);
        ButterKnife.bind(this);

        ordersRepository = Inject.provideOrdersRepository();

        extractExtras();
        setupToolbar();
    }

    @Override
    protected void onStart() {
        super.onStart();

        setupFab();
        setupOrdersList();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == REQUEST_CODE_ORDER_PROCESS) {
            switch (resultCode) {
                case RESULT_CODE_OK:
                case RESULT_CODE_SKIP:
                    ++current;
                    break;
                case RESULT_CODE_OK_UNIQUE:
                case RESULT_CODE_CLOSE:
                default:
                    current = -1;
            }

            processNext();
        }
    }

    private void extractExtras() {
        Bundle extra = getIntent().getExtras();
        if (extra != null) {
            filterList = extra.getParcelable(EXTRA_ORDER_FILTERS);
            processAll = extra.getBoolean(EXTRA_PROCESS_ALL, false);
        }
    }

    private void setupToolbar() {
        toolbar.setNavigationIcon(R.drawable.ic_back_white_vector);
```

```
        toolbar.setNavigationOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                onBackPressed();
            }
        });
    }

    private void setupOrdersList() {
        current = 0;

        ordersRepository.getList(
                filterList,
                new RepositoryCallback<List<Order>>() {
                    @Override
                    public void onSuccess(List<Order> result) {
                        orders = result;
                        showOrdersList();
                    }

                    @Override
                    public void onError(List<String> errors) {
                        if(errors != null) {
                            showError(errors.get(0));
                        } else {
                            showCommunicationError();
                        }
                    }
                });
    }

    private void setupFab() {
        if (processAll) {
            fab.setVisibility(View.VISIBLE);
        } else {
            fab.setVisibility(View.GONE);
        }
    }

    private void showOrdersList() {
        if(orders.size() == 0) {
            showEmptyList();
            return;
        }

        recyclerView.setAdapter(
                new OrdersListAdapter(
                        this,
                        orders,
                        new OrdersListAdapter.OnClickListener() {
                            @Override
                            public void onClick(int orderId) {
                                process(orderId, 1, 1);
                            }
                        }));
    }

    private void showEmptyList() {
        recyclerView.setAdapter(
                new EmptyStateAdapter(
```

```
                        this,
                        R.drawable.ic_coffee_black_vector,
                        getString(R.string.order_list_empty)));
    }

    private void showCommunicationError() {
        showError(
                getString(R.string.error_communication));
    }

    private void showError(String error) {
        recyclerView.setAdapter(
                new EmptyStateAdapter(
                        this,
                        error));
    }

    private void processNext() {
        if (current >= orders.size() ||
                current < 0) {
            setupOrdersList();
            return;
        }

        process(orders.get(current).id(),
                current + 1,
                orders.size());
    }

    private void process(int orderId, int position, int total) {
        Intent intent =
                OrderDetailsActivity.getIntentForOrder(
                        this,
                        orderId,
                        position,
                        total);

        startActivityForResult(intent, REQUEST_CODE_ORDER_PROCESS);
    }

    @OnClick(R.id.orders_list_fab)
    void onFabClick() {
        processNext();
    }
}
```

Listagem B.63 – OrdersListAdapter

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list;

import android.content.Context;
import android.support.v4.content.ContextCompat;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import java.util.List;
```

```
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by andre on 10/07/17.
 */
final class OrdersListAdapter extends RecyclerView.Adapter<OrdersListAdapter.ViewHolder> {

    private final Context context;
    private final List<Order> orders;
    private final LayoutInflater layoutInflater;
    private final OnClickListener listener;

    OrdersListAdapter(
            Context context,
            List<Order> orders,
            OnClickListener listener) {

        this.context = context;
        this.orders = orders;
        this.listener = listener;

        layoutInflater = LayoutInflater.from(context);
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        return new ViewHolder(
                layoutInflater.inflate(
                        R.layout.row_orders_list,
                        parent,
                        false));
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, final int position) {
        final Order order = orders.get(position);
        boolean isProcessed = order.isProcessed();

        holder.id.setText(
                String.valueOf(
                        order.id()));

        holder.shipType.setText(order.shipmentInfo().shipType());

        holder.itemsCount.setText(
                String.valueOf(
                        order.size()));

        holder.lastModifiedAt.setText(order.lastModifiedAt());

        holder.processedAt.setTextColor(
                ContextCompat.getColor(
                        context,
                        isProcessed ? R.color.green : R.color.red));

        holder.processedAt.setText(getProcessedAt(order));
```

```
        holder.action.setText(
                context.getString(
                        isProcessed ? R.string.orders_list_action_details : R.string.orders_list_acti

        holder.root.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        listener.onClick(order.id());
                    }
                });
    }

    @Override
    public int getItemCount() {
        return orders.size();
    }

    private String getProcessedAt(Order order) {
        if (order.isProcessed()) {
            return order.processedAt();
        }

        return context.getString(R.string.order_list_processed_at_empty);
    }

    static class ViewHolder extends RecyclerView.ViewHolder {

        @BindView(R.id.order_root)
        View root;

        @BindView(R.id.order_id_value)
        TextView id;

        @BindView(R.id.order_ship_type_value)
        TextView shipType;

        @BindView(R.id.order_size_value)
        TextView itemsCount;

        @BindView(R.id.order_processed_at_value)
        TextView processedAt;

        @BindView(R.id.order_last_modification_date_value)
        TextView lastModifiedAt;

        @BindView(R.id.order_action_text)
        TextView action;

        ViewHolder(View itemView) {
            super(itemView);

            ButterKnife.bind(this, itemView);
        }
    }

    interface OnClickListener {
        void onClick(int orderId);
    }
}
```

B.2.1 Testes

Listagem B.64 – BarcodeProcessorActivityTest

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;

import android.app.AlertDialog;
import android.content.Intent;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.MockitoAnnotations;
import org.robolectric.Robolectric;
import org.robolectric.RobolectricTestRunner;
import org.robolectric.annotation.Config;
import org.robolectric.shadows.ShadowActivity;
import org.robolectric.shadows.ShadowAlertDialog;
import org.robolectric.shadows.ShadowLooper;

import java.util.HashMap;
import java.util.Map;

import br.com.aaascp.gerenciadordepedidos.BuildConfig;
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;

import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertTrue;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.containsString;
import static org.robolectric.Shadows.shadowOf;

/**
 * Created by andre on 09/10/17.
 */
@RunWith(RobolectricTestRunner.class)
@Config(constants = BuildConfig.class)
public class BarcodeProcessorActivityTest {
    private static final int ORDER_ID = 1000;
    private static final int CODE_BASE = 1000;
    private static final String CODE_INVALID = "CODE_INVALID";

    private BarcodeProcessorActivity activity;

    private CodesToProcess codesToProcess;

    @Before
    public void setup() {
        MockitoAnnotations.initMocks(this);
    }

    private void init(int... left) {
        Map<String, Integer> codes = new HashMap<>();

        for (int i = 0; i < left.length; i++) {
            codes.put(String.valueOf(CODE_BASE + i), left[i]);
        }

        codesToProcess = CodesToProcess.create(codes, ORDER_ID);
```

```
        Intent intent = new Intent();
        intent.putExtra(
                BarcodeProcessorActivity.EXTRA_CODES_TO_PROCESS,
                codesToProcess);
        intent.putExtra(
                BarcodeProcessorActivity.EXTRA_ORDER_ID,
                ORDER_ID);

        activity = Robolectric.buildActivity(BarcodeProcessorActivity.class, intent)
                .create()
                .start()
                .get();
    }

    @Test
    public void onCreate_setupToolbar() {
        init();

        assertThat(
                activity.title.getText().toString(),
                containsString(String.valueOf(ORDER_ID)));
    }

    @Test
    public void onCreate_setItemsLeft() {
        init(1, 2);

        assertThat(
                activity.itemsLeft.getText().toString(),
                containsString("3"));
    }

    @Test
    public void onCreate_setItemsLeft_finished() {
        init(0);

        assertThat(
                activity.itemsLeft.getText().toString(),
                containsString("0"));
    }

    @Test
    public void onItemProcessed_success_changesItemsLeft() {
        init(2);

        activity.onItemProcessed(String.valueOf(CODE_BASE));

        assertThat(
                activity.itemsLeft.getText().toString(),
                containsString("1"));
    }

    @Test
    public void onItemProcessed_success_finish() {
        init(1);

        activity.onItemProcessed(String.valueOf(CODE_BASE));

        AlertDialog dialog = ShadowAlertDialog.getLatestAlertDialog();
```

```
        ShadowAlertDialog shadowDialog = shadowOf(dialog);

        assertEquals(
                shadowDialog.getTitle().toString(),
                activity.getString(R.string.barcode_processor_finish_dialog_title));
        assertEquals(
                shadowDialog.getMessage().toString(),
                activity.getString(R.string.barcode_processor_finish_dialog_message));
    }


    @Test
    public void onItemProcessed_alreadyProcessed_notChangesItemsLeft() {
        init(1);

        activity.onItemProcessed(String.valueOf(CODE_BASE));
        ShadowLooper.runUiThreadTasksIncludingDelayedTasks();
        activity.onItemProcessed(String.valueOf(CODE_BASE));

        assertThat(
                activity.itemsLeft.getText().toString(),
                containsString("0"));
    }

    @Test
    public void onItemProcessed_codeInvalid_notChangesItemsLeft() {
        init(1);

        activity.onItemProcessed(String.valueOf(CODE_INVALID));

        assertThat(
                activity.itemsLeft.getText().toString(),
                containsString("1"));
    }

    @Test
    public void onItemProcessed_notReady_notProcesses() {
        init(2);

        activity.onItemProcessed(String.valueOf(CODE_BASE));
        activity.onItemProcessed(String.valueOf(CODE_BASE));

        assertThat(
                activity.itemsLeft.getText().toString(),
                containsString("1"));
    }

    @Test
    public void onClose_setsResult() {
        init(1, 1);

        activity.onItemProcessed(String.valueOf(CODE_BASE));
        activity.finish();

        ShadowActivity shadowActivity = shadowOf(activity);

        assertEquals(
                BarcodeProcessorActivity.RESULT_OK,
                shadowActivity.getResultCode());
        assertEquals(
```

```
                shadowActivity.getResultIntent().getParcelableExtra(
                        BarcodeProcessorActivity.EXTRA_RESULT),
                codesToProcess);

        assertTrue(shadowActivity.isFinishing());
    }

    @Test
    public void onFinish_setsResult() {
        init(1);

        activity.onItemProcessed(String.valueOf(CODE_BASE));
        activity.finish();

        ShadowActivity shadowActivity = shadowOf(activity);

        assertEquals(
                BarcodeProcessorActivity.RESULT_OK,
                shadowActivity.getResultCode());
        assertEquals(
                shadowActivity.getResultIntent().getParcelableExtra(
                        BarcodeProcessorActivity.EXTRA_RESULT),
                codesToProcess);

        assertTrue(shadowActivity.isFinishing());
    }
}
```

Listagem B.65 – OrderDetailsActivityTest

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.details;

import android.Manifest;
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
import android.view.MenuItem;
import android.view.View;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.MockitoAnnotations;
import org.robolectric.Robolectric;
import org.robolectric.RobolectricTestRunner;
import org.robolectric.annotation.Config;
import org.robolectric.fakes.RoboMenuItem;
import org.robolectric.shadows.ShadowActivity;
import org.robolectric.shadows.ShadowAlertDialog;

import java.util.HashMap;
import java.util.Map;

import br.com.aaascp.gerenciadordepedidos.BuildConfig;
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderItem;
import br.com.aaascp.gerenciadordepedidos.factory.OrdersFactory;
```

```
import br.com.aaascp.gerenciadordepedidos.presentation.ui.camera.BarcodeProcessorActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.info.OrderInfoActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.item.OrderItemActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list.OrdersListActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.EmptyStateAdapter;
import br.com.aaascp.gerenciadordepedidos.repository.dao.OrdersFakeDataSource;

import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.containsString;
import static org.junit.Assert.assertEquals;
import static org.junit.Assert.assertTrue;
import static org.robolectric.Shadows.shadowOf;

/**
 * Created by andre on 02/10/17.
 */

@RunWith(RobolectricTestRunner.class)
@Config(constants = BuildConfig.class)
public class OrderDetailsActivityTest {

    private static final Order ORDER_NOT_FINISHED =
            OrdersFactory.createOrder(1000, false);

    private static final Order ORDER_FINISHED =
            OrdersFactory.createOrder(1001, true);

    private static final Order ORDER_ERROR =
            OrdersFakeDataSource.createOrderError();

    private static final Order ORDER_COMMUNICATION_ERROR =
            OrdersFakeDataSource.createOrderCommunicationError();

    private OrderDetailsActivity activity;

    @Before
    public void setup() {
        MockitoAnnotations.initMocks(this);
    }

    private void init(Order order, int current, int total) {
        OrdersFakeDataSource.clear();
        OrdersFakeDataSource.addOrder(order);

        Intent intent = new Intent();
        intent.putExtra(
                OrderDetailsActivity.EXTRA_ORDER_ID,
                order.id());
        intent.putExtra(
                OrderDetailsActivity.EXTRA_TOTAL,
                total);
        intent.putExtra(
                OrderDetailsActivity.EXTRA_CURRENT,
                current);

        activity = Robolectric.buildActivity(OrderDetailsActivity.class, intent)
                .create()
                .start()
                .get();
```

```
        activity.recyclerView.measure(0, 0);
        activity.recyclerView.layout(0, 0, 100, 1000);
    }

    private void init(Order order) {
        init(order, 1, 1);
    }

    @Test
    public void setupTitle_unique() throws Exception {
        int current = 1;
        int total = 1;
        init(ORDER_FINISHED, current, total);

        assertThat(
                activity.toolbar.getTitle().toString(),
                containsString(String.valueOf(ORDER_FINISHED.id())));
    }

    @Test
    public void setupTitle_processAll() throws Exception {
        int current = 1;
        int total = 2;
        init(ORDER_FINISHED, current, total);

        assertThat(
                activity.toolbar.getTitle().toString(),
                containsString(
                        String.format("%d (%d/%d)", ORDER_FINISHED.id(),
                                current,
                                total)));
    }

    @Test
    public void setShipType() throws Exception {
        init(ORDER_FINISHED);

        assertEquals(
                activity.shipTypeView.getValue(),
                ORDER_FINISHED.shipmentInfo().shipType());
    }

    @Test
    public void setItemsLeft() throws Exception {
        init(ORDER_NOT_FINISHED);

        assertEquals(
                activity.itemsLeftView.getText().toString(),
                "0 / 2");
    }

    @Test
    public void showOrder() throws Exception {
        init(ORDER_FINISHED);

        OrderDetailsAdapter.ViewHolder holder;
        int i = 0;
        for (OrderItem item : ORDER_FINISHED.items().values()) {
            holder = (OrderDetailsAdapter.ViewHolder) activity.recyclerView
                    .findViewHolderForAdapterPosition(i);
```

```
            int itemsLeft =
                    item.quantity() - ORDER_FINISHED.codesToProcess().codes().get(item.code());

            String itemsLeftText =
                    String.format(
                            activity.getString(R.string.order_details_count_text),
                            itemsLeft,
                            item.quantity());

            assertEquals(
                    holder.code.getText().toString(),
                    item.code());
            assertEquals(
                    holder.quantity.getText().toString(),
                    itemsLeftText);
            assertEquals(
                    holder.description.getText().toString(),
                    item.description());

            i++;
        }
    }

    @Test
    public void showError() throws Exception {
        init(ORDER_ERROR);

        EmptyStateAdapter.ViewHolder holder =
                (EmptyStateAdapter.ViewHolder) activity.recyclerView
                        .findViewHolderForAdapterPosition(0);

        assertEquals(
                holder.getMessage().getText().toString(),
                OrdersFakeDataSource.ERROR_MESSAGE);
    }

    @Test
    public void showCommunicationError() throws Exception {
        init(ORDER_COMMUNICATION_ERROR);

        EmptyStateAdapter.ViewHolder holder =
                (EmptyStateAdapter.ViewHolder) activity.recyclerView
                        .findViewHolderForAdapterPosition(0);

        String message = activity.getString(R.string.error_communication);
        assertEquals(
                holder.getMessage().getText().toString(),
                message);
    }

    @Test
    public void showItemsLeftLabel() throws Exception {
        init(ORDER_NOT_FINISHED);

        assertEquals(
                activity.itemsLeftRoot.getVisibility(),
                View.VISIBLE);
    }
```

```
@Test
public void showAlreadyProcessedLabel() throws Exception {
    init(ORDER_FINISHED);

    assertEquals(
            activity.alreadyProcessedRoot.getVisibility(),
            View.VISIBLE);
}

@Test
public void navigateToDetails() throws Exception {
    init(ORDER_FINISHED);

    OrderDetailsAdapter.ViewHolder holder =
            (OrderDetailsAdapter.ViewHolder) activity.recyclerView
                    .findViewHolderForAdapterPosition(0);

    holder.root.performClick();

    ShadowActivity shadowActivity = shadowOf(activity);
    Intent intent = shadowActivity.getNextStartedActivity();

    OrderItem item = ORDER_FINISHED.items().get(holder.code.getText().toString());
    assertEquals(intent.getParcelableExtra(OrderItemActivity.EXTRA_ITEM), item);
}

@Test
public void finishOk_save() throws Exception {
    init(ORDER_NOT_FINISHED);

    activity.onFinishedClick();
    Order order = OrdersFakeDataSource.load(ORDER_NOT_FINISHED.id());

    assertTrue(order.isProcessed());
}

@Test
public void finishOk_list() throws Exception {
    init(ORDER_FINISHED, 1, 2);

    activity.onFinishedClick();
    ShadowActivity shadowActivity = shadowOf(activity);

    assertEquals(
            OrdersListActivity.RESULT_CODE_OK,
            shadowActivity.getResultCode());

    assertTrue(shadowActivity.isFinishing());
}

@Test
public void finishOk_unique() throws Exception {
    init(ORDER_FINISHED, 1, 1);

    activity.onFinishedClick();
    ShadowActivity shadowActivity = shadowOf(activity);

    assertEquals(
            OrdersListActivity.RESULT_CODE_OK_UNIQUE,
            shadowActivity.getResultCode());
```

```
        assertTrue(shadowActivity.isFinishing());
    }

    @Test
    public void onBackPressed_orderFinished_finish() throws Exception {
        init(ORDER_FINISHED);

        activity.onBackPressed();
        assertTrue(shadowOf(activity).isFinishing());
    }

    @Test
    public void onBackPressed_orderNotFinished_showBackDialog() throws Exception {
        init(ORDER_NOT_FINISHED);
        activity.onBackPressed();

        AlertDialog dialog = ShadowAlertDialog.getLatestAlertDialog();
        ShadowAlertDialog shadowDialog = shadowOf(dialog);

        assertEquals(
                shadowDialog.getTitle().toString(),
                activity.getString(R.string.order_details_back_dialog_title));
        assertEquals(
                shadowDialog.getMessage().toString(),
                activity.getString(R.string.order_details_back_dialog_message));
    }

    @Test
    public void showBackDialogOk_finish() throws Exception {
        init(ORDER_NOT_FINISHED);
        activity.onBackPressed();

        AlertDialog dialog = ShadowAlertDialog.getLatestAlertDialog();
        dialog.getButton(DialogInterface.BUTTON_POSITIVE).performClick();

        assertTrue(shadowOf(activity).isFinishing());
    }

    @Test
    public void showSkipDialogOk_finishSkip() throws Exception {
        init(ORDER_FINISHED, 1, 2);

        ShadowActivity shadowActivity = shadowOf(activity);
        MenuItem menuItem = new RoboMenuItem(R.id.menu_order_details_skip);
        activity.onOptionsItemSelected(menuItem);

        AlertDialog dialog = ShadowAlertDialog.getLatestAlertDialog();
        dialog.getButton(DialogInterface.BUTTON_POSITIVE).performClick();

        assertEquals(
                OrdersListActivity.RESULT_CODE_SKIP,
                shadowActivity.getResultCode());

        assertTrue(shadowActivity.isFinishing());
    }

    @Test
    public void showClearDialogOk_refreshOrder() throws Exception {
        init(ORDER_NOT_FINISHED);
```

```
        MenuItem menuItem = new RoboMenuItem(R.id.menu_order_details_clear);
        activity.onOptionsItemSelected(menuItem);

        AlertDialog dialog = ShadowAlertDialog.getLatestAlertDialog();
        dialog.getButton(DialogInterface.BUTTON_POSITIVE).performClick();

        shadowOf(activity).grantPermissions(Manifest.permission.CAMERA);
        activity.onFabClick();

        ShadowActivity.IntentForResult intentResult =
                shadowOf(activity).getNextStartedActivityForResult();

        Map<String, Integer> codes = new HashMap<>();
        for (String code : ORDER_NOT_FINISHED.codesToProcess().codes().keySet()) {
            codes.put(code, 0);
        }
        CodesToProcess newCodesToProcess =
                CodesToProcess.create(codes, ORDER_NOT_FINISHED.id());

        Intent result = new Intent();
        result.putExtra(BarcodeProcessorActivity.EXTRA_RESULT, newCodesToProcess);

        shadowOf(activity).receiveResult(
                intentResult.intent,
                Activity.RESULT_OK,
                result);

        activity.recyclerView.measure(0, 0);
        activity.recyclerView.layout(0, 0, 100, 1000);

        OrderDetailsAdapter.ViewHolder holder;
        int i = 0;
        for (OrderItem item : ORDER_NOT_FINISHED.items().values()) {
            holder = (OrderDetailsAdapter.ViewHolder) activity.recyclerView
                    .findViewHolderForAdapterPosition(i);

            int itemsLeft =
                    item.quantity() - newCodesToProcess.codes().get(item.code());

            String itemsLeftText =
                    String.format(
                            activity.getString(R.string.order_details_count_text),
                            itemsLeft,
                            item.quantity());

            assertEquals(
                    holder.code.getText().toString(),
                    item.code());
            assertEquals(
                    holder.quantity.getText().toString(),
                    itemsLeftText);
            assertEquals(
                    holder.description.getText().toString(),
                    item.description());

            i++;
        }
    }
```

```
@Test
public void onMoreDetailsClick_showDetails() throws Exception {
    init(ORDER_FINISHED);

    MenuItem menuItem =
            new RoboMenuItem(R.id.menu_order_details_more_details);
    activity.onOptionsItemSelected(menuItem);

    Intent startedIntent = shadowOf(activity).getNextStartedActivity();

    assertEquals(
            startedIntent.getParcelableExtra(OrderInfoActivity.EXTRA_ORDER),
            ORDER_FINISHED);
}

@Test
public void onClearClick_showClearDialog() throws Exception {
    init(ORDER_FINISHED);

    MenuItem menuItem = new RoboMenuItem(R.id.menu_order_details_clear);
    activity.onOptionsItemSelected(menuItem);

    AlertDialog dialog = ShadowAlertDialog.getLatestAlertDialog();
    ShadowAlertDialog shadowDialog = shadowOf(dialog);

    assertEquals(
            shadowDialog.getTitle().toString(),
            activity.getString(R.string.order_details_clear_dialog_title));
    assertEquals(
            shadowDialog.getMessage().toString(),
            activity.getString(R.string.order_details_clear_dialog_message));
}

@Test
public void onSkipClick_showSkipDialog() throws Exception {
    init(ORDER_FINISHED);

    MenuItem menuItem = new RoboMenuItem(R.id.menu_order_details_skip);
    activity.onOptionsItemSelected(menuItem);

    AlertDialog dialog = ShadowAlertDialog.getLatestAlertDialog();
    ShadowAlertDialog shadowDialog = shadowOf(dialog);

    assertEquals(
            shadowDialog.getTitle().toString(),
            activity.getString(R.string.order_details_skip_dialog_title));
    assertEquals(
            shadowDialog.getMessage().toString(),
            activity.getString(R.string.order_details_skip_dialog_message));
}

@Test
public void onFinishedClick_finishOk_list() throws Exception {
    init(ORDER_FINISHED, 1, 2);

    ShadowActivity shadowActivity = shadowOf(activity);
    View onFinishedClick = activity.findViewById(R.id.order_details_finish);
    onFinishedClick.performClick();

    assertEquals(
```

```
                OrdersListActivity.RESULT_CODE_OK,
                shadowActivity.getResultCode());

        assertTrue(shadowActivity.isFinishing());
    }

    @Test
    public void onFinishedClick_finishOk_unique() throws Exception {
        init(ORDER_FINISHED, 1, 1);

        ShadowActivity shadowActivity = shadowOf(activity);
        View onFinishedClick = activity.findViewById(R.id.order_details_finish);
        onFinishedClick.performClick();

        assertEquals(
                OrdersListActivity.RESULT_CODE_OK_UNIQUE,
                shadowActivity.getResultCode());

        assertTrue(shadowActivity.isFinishing());
    }

    @Test
    public void onAlreadyProcessedClick_finishClose() throws Exception {
        init(ORDER_FINISHED, 1, 2);

        ShadowActivity shadowActivity = shadowOf(activity);
        View onAlreadyProcessedClick = activity.findViewById(R.id.order_details_processed);
        onAlreadyProcessedClick.performClick();

        assertEquals(
                OrdersListActivity.RESULT_CODE_CLOSE,
                shadowActivity.getResultCode());

        assertTrue(shadowActivity.isFinishing());
    }

    @Test
    public void onShipTypeClick_showDetails() throws Exception {
        init(ORDER_FINISHED);

        View onShipTypeClick = activity.findViewById(R.id.order_details_ship_type);
        onShipTypeClick.performClick();

        Intent startedIntent = shadowOf(activity).getNextStartedActivity();

        assertEquals(
                startedIntent.getParcelableExtra(OrderInfoActivity.EXTRA_ORDER),
                ORDER_FINISHED);
    }

    @Test
    public void onFabClick_navigateToProcessor() throws Exception {
        init(ORDER_NOT_FINISHED);

        shadowOf(activity).grantPermissions(Manifest.permission.CAMERA);

        activity.onFabClick();

        ShadowActivity shadowActivity = shadowOf(activity);
        ShadowActivity.IntentForResult intentResult =
```

```
            shadowActivity.getNextStartedActivityForResult();

    assertEquals(
            intentResult.intent.getParcelableExtra(
                    BarcodeProcessorActivity.EXTRA_CODES_TO_PROCESS),
            ORDER_NOT_FINISHED.codesToProcess());

    assertEquals(intentResult.requestCode, OrderDetailsActivity.REQUEST_CODE_PROCESS);
}

@Test
public void onActivityResult_refreshCodesProcessed() throws Exception {
    init(ORDER_NOT_FINISHED);

    shadowOf(activity).grantPermissions(Manifest.permission.CAMERA);
    activity.onFabClick();

    ShadowActivity.IntentForResult intentResult =
            shadowOf(activity).getNextStartedActivityForResult();

    Map<String, Integer> codes = new HashMap<>();
    for (String code : ORDER_NOT_FINISHED.codesToProcess().codes().keySet()) {
        codes.put(code, 0);
    }
    CodesToProcess newCodesToProcess =
            CodesToProcess.create(codes, ORDER_NOT_FINISHED.id());

    Intent result = new Intent();
    result.putExtra(BarcodeProcessorActivity.EXTRA_RESULT, newCodesToProcess);

    shadowOf(activity).receiveResult(
            intentResult.intent,
            Activity.RESULT_OK,
            result);

    assertEquals(
            intentResult.requestCode,
            OrderDetailsActivity.REQUEST_CODE_PROCESS);

    activity.recyclerView.measure(0, 0);
    activity.recyclerView.layout(0, 0, 100, 1000);

    OrderDetailsAdapter.ViewHolder holder;
    int i = 0;
    for (OrderItem item : ORDER_NOT_FINISHED.items().values()) {
        holder = (OrderDetailsAdapter.ViewHolder) activity.recyclerView
                .findViewHolderForAdapterPosition(i);

        int itemsLeft =
                item.quantity() - newCodesToProcess.codes().get(item.code());

        String itemsLeftText =
                String.format(
                        activity.getString(R.string.order_details_count_text),
                        itemsLeft,
                        item.quantity());

        assertEquals(
                holder.code.getText().toString(),
                item.code());
```

```
            assertEquals(
                    holder.quantity.getText().toString(),
                    itemsLeftText);
            assertEquals(
                    holder.description.getText().toString(),
                    item.description());

            i++;
        }
    }

    @Test
    public void onShowOrder_alreadyProcessed() throws Exception {
        init(ORDER_FINISHED);

        assertEquals(
                activity.alreadyProcessedRoot.getVisibility(),
                View.VISIBLE);
        assertEquals(
                activity.itemsLeftRoot.getVisibility(),
                View.GONE);
        assertEquals(
                activity.finishRoot.getVisibility(),
                View.GONE);
    }

    @Test
    public void onShowOrder_finished() throws Exception {
        init(ORDER_NOT_FINISHED);

        shadowOf(activity).grantPermissions(Manifest.permission.CAMERA);
        activity.onFabClick();

        ShadowActivity.IntentForResult intentResult =
                shadowOf(activity).getNextStartedActivityForResult();

        Map<String, Integer> codes = new HashMap<>();
        for (String code : ORDER_NOT_FINISHED.codesToProcess().codes().keySet()) {
            codes.put(code, 0);
        }
        CodesToProcess newCodesToProcess =
                CodesToProcess.create(codes, ORDER_NOT_FINISHED.id());

        Intent result = new Intent();
        result.putExtra(
                BarcodeProcessorActivity.EXTRA_RESULT,
                newCodesToProcess);

        shadowOf(activity).receiveResult(
                intentResult.intent,
                Activity.RESULT_OK,
                result);

        assertEquals(
                activity.finishRoot.getVisibility(),
                View.VISIBLE);
        assertEquals(
                activity.alreadyProcessedRoot.getVisibility(),
                View.GONE);
        assertEquals(
```

```
                activity.itemsLeftRoot.getVisibility(),
                View.GONE);
    }

    @Test
    public void onShowOrder_showItemsLeft() throws Exception {
        init(ORDER_NOT_FINISHED);

        assertEquals(
                activity.itemsLeftRoot.getVisibility(),
                View.VISIBLE);
        assertEquals(
                activity.alreadyProcessedRoot.getVisibility(),
                View.GONE);
        assertEquals(
                activity.finishRoot.getVisibility(),
                View.GONE);
    }
}
```

Listagem B.66 – OrdersListActivityTest

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list;

import android.content.Intent;
import android.support.v4.content.ContextCompat;
import android.view.View;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.MockitoAnnotations;
import org.robolectric.Robolectric;
import org.robolectric.RobolectricTestRunner;
import org.robolectric.annotation.Config;
import org.robolectric.shadows.ShadowActivity;

import java.util.Arrays;
import java.util.Collections;
import java.util.List;

import br.com.aaascp.gerenciadordepedidos.BuildConfig;
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.NullOrderFilterList;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.factory.OrdersFactory;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.details.OrderDetailsActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.EmptyStateAdapter;
import br.com.aaascp.gerenciadordepedidos.repository.dao.OrdersFakeDataSource;

import static org.junit.Assert.assertEquals;
import static org.robolectric.Shadows.shadowOf;

/**
 * Created by andre on 02/10/17.
 */

@RunWith(RobolectricTestRunner.class)
@Config(constants = BuildConfig.class)
public class OrdersListActivityTest {
```

```
private static Order ORDER_PROCESSED =
        OrdersFactory.createOrder(1000, true);

private static Order ORDER_NOT_PROCESSED =
        OrdersFactory.createOrder(1001, false);

private static List<Order> ORDERS =
        Arrays.asList(ORDER_PROCESSED, ORDER_NOT_PROCESSED);

private static List<Order> EMPTY_LIST = Collections.emptyList();

private static List<Order> ERRORS =
        Collections.singletonList(
                OrdersFakeDataSource.createOrderError());

private static List<Order> COMMUNICATION_ERROR =
        Collections.singletonList(
                OrdersFakeDataSource.createOrderCommunicationError());

private OrdersListActivity activity;

@Before
public void setup() throws Exception {
    MockitoAnnotations.initMocks(this);
}

private void init(boolean processAll) {
    Intent intent = new Intent();
    intent.putExtra(
            OrdersListActivity.EXTRA_PROCESS_ALL, processAll);
    intent.putExtra(OrdersListActivity.EXTRA_ORDER_FILTERS,
            NullOrderFilterList.create());

    activity = Robolectric.buildActivity(OrdersListActivity.class, intent)
            .create()
            .start()
            .get();
}

private void init(List<Order> orders) {
    OrdersFakeDataSource.clear();

    for (Order order : orders) {
        OrdersFakeDataSource.addOrder(order);
    }

    init(false);

    activity.recyclerView.measure(0, 0);
    activity.recyclerView.layout(0, 0, 100, 1000);
}

@Test
public void setupFab_processAll() throws Exception {
    init(true);

    assertEquals(activity.fab.getVisibility(), View.VISIBLE);
}
```

```
@Test
public void setupFab_notProcessAll() throws Exception {
    init(false);

    assertEquals(activity.fab.getVisibility(), View.GONE);
}

@Test
public void setupOrdersList_success() throws Exception {
    init(ORDERS);

    OrdersListAdapter.ViewHolder holder =
            (OrdersListAdapter.ViewHolder) activity.recyclerView
                    .findViewHolderForAdapterPosition(0);

    assertEquals(
            holder.id.getText().toString(),
            String.valueOf(ORDERS.get(0).id()));
    assertEquals(
            holder.shipType.getText().toString(),
            ORDERS.get(0).shipmentInfo().shipType());
    assertEquals(
            holder.itemsCount.getText().toString(),
            String.valueOf(ORDERS.get(0).size()));
    assertEquals(
            holder.processedAt.getText().toString(),
            ORDERS.get(0).processedAt());
    assertEquals(
            holder.processedAt.getCurrentTextColor(),
            ContextCompat.getColor(activity, R.color.green));
    assertEquals(
            holder.lastModifiedAt.getText().toString(),
            ORDERS.get(0).lastModifiedAt());
    assertEquals(
            holder.action.getText().toString(),
            String.valueOf(activity.getString(R.string.orders_list_action_details)));

    holder = (OrdersListAdapter.ViewHolder) activity.recyclerView
            .findViewHolderForAdapterPosition(1);

    assertEquals(
            holder.id.getText().toString(),
            String.valueOf(ORDERS.get(1).id()));
    assertEquals(
            holder.shipType.getText().toString(),
            ORDERS.get(1).shipmentInfo().shipType());
    assertEquals(
            holder.itemsCount.getText().toString(),
            String.valueOf(ORDERS.get(1).size()));
    assertEquals(
            holder.processedAt.getText().toString(),
            activity.getString(R.string.order_list_processed_at_empty));
    assertEquals(
            holder.processedAt.getCurrentTextColor(),
            ContextCompat.getColor(activity, R.color.red));
    assertEquals(
            holder.lastModifiedAt.getText().toString(),
            ORDERS.get(1).lastModifiedAt());
    assertEquals(
            holder.action.getText().toString(),
```

```
            String.valueOf(activity.getString(R.string.orders_list_action_process)));
    }

    @Test
    public void setupOrdersList_empty() throws Exception {
        init(EMPTY_LIST);

        EmptyStateAdapter.ViewHolder holder =
                (EmptyStateAdapter.ViewHolder) activity.recyclerView
                        .findViewHolderForAdapterPosition(0);

        assertEquals(
                holder.getMessage().getText().toString(),
                activity.getString(R.string.order_list_empty));
    }

    @Test
    public void setupOrdersList_error() throws Exception {
        init(ERRORS);

        EmptyStateAdapter.ViewHolder holder =
                (EmptyStateAdapter.ViewHolder) activity.recyclerView
                        .findViewHolderForAdapterPosition(0);

        assertEquals(
                holder.getMessage().getText().toString(),
                OrdersFakeDataSource.ERROR_MESSAGE);
    }

    @Test
    public void setupOrdersList_communicationError() throws Exception {
        init(COMMUNICATION_ERROR);

        EmptyStateAdapter.ViewHolder holder =
                (EmptyStateAdapter.ViewHolder) activity.recyclerView
                        .findViewHolderForAdapterPosition(0);

        assertEquals(
                holder.getMessage().getText().toString(),
                activity.getString(R.string.error_communication));
    }

    @Test
    public void processAll_startsDetailsForListWithFirstOrder() throws Exception {
        init(ORDERS);

        activity.fab.performClick();

        Intent startedIntent = shadowOf(activity).getNextStartedActivity();
        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_TOTAL),
                ORDERS.size());
        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_CURRENT),
                1);
        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_ORDER_ID),
                ORDERS.get(0).id());
    }
```

```
@Test
public void processAll_resultOk_startsDetailsForNext() throws Exception {
    init(ORDERS);

    activity.fab.performClick();

    ShadowActivity shadowActivity = shadowOf(activity);

    Intent startedIntent = shadowActivity.getNextStartedActivity();

    shadowActivity.receiveResult(
            startedIntent,
            OrdersListActivity.RESULT_CODE_OK,
            null);

    Intent nextIntent = shadowActivity.getNextStartedActivity();

    assertEquals(
            nextIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_TOTAL),
            ORDERS.size());
    assertEquals(
            nextIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_CURRENT),
            2);
    assertEquals(
            nextIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_ORDER_ID),
            ORDERS.get(1).id());
}

@Test
public void processAll_resultOk_last_resetsCurrent() throws Exception {
    init(ORDERS);

    activity.fab.performClick();

    ShadowActivity shadowActivity = shadowOf(activity);

    for (int i = 0; i < ORDERS.size(); i++) {
        shadowActivity.receiveResult(
                shadowActivity.getNextStartedActivity(),
                OrdersListActivity.RESULT_CODE_OK,
                null);
    }

    activity.fab.performClick();

    Intent startedIntent = shadowActivity.getNextStartedActivity();
    assertEquals(
            startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_TOTAL),
            ORDERS.size());
    assertEquals(
            startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_CURRENT),
            1);
    assertEquals(
            startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_ORDER_ID),
            ORDERS.get(0).id());
}

@Test
public void processAl_resultSkip_startsDetailsForNext() throws Exception {
    init(ORDERS);
```

```
        activity.fab.performClick();

        ShadowActivity shadowActivity = shadowOf(activity);

        Intent startedIntent = shadowActivity.getNextStartedActivity();

        shadowActivity.receiveResult(
                startedIntent,
                OrdersListActivity.RESULT_CODE_SKIP,
                null);

        Intent nextIntent = shadowActivity.getNextStartedActivity();

        assertEquals(
                nextIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_TOTAL),
                ORDERS.size());
        assertEquals(
                nextIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_CURRENT),
                2);
        assertEquals(
                nextIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_ORDER_ID),
                ORDERS.get(1).id());
    }

    @Test
    public void processAll_resultSkip_last_resetsCurent() throws Exception {
        init(ORDERS);

        activity.fab.performClick();

        ShadowActivity shadowActivity = shadowOf(activity);

        for (int i = 0; i < ORDERS.size(); i++) {
            shadowActivity.receiveResult(
                    shadowActivity.getNextStartedActivity(),
                    OrdersListActivity.RESULT_CODE_SKIP,
                    null);
        }

        activity.fab.performClick();

        Intent startedIntent = shadowActivity.getNextStartedActivity();
        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_TOTAL),
                ORDERS.size());
        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_CURRENT),
                1);
        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_ORDER_ID),
                ORDERS.get(0).id());
    }

    @Test
    public void processAll_resultClose_resetsCurrent() throws Exception {
        init(ORDERS);

        activity.fab.performClick();
```

```
        ShadowActivity shadowActivity = shadowOf(activity);

        shadowActivity.receiveResult(
                shadowActivity.getNextStartedActivity(),
                OrdersListActivity.RESULT_CODE_CLOSE,
                null);

        activity.fab.performClick();
        Intent nextIntent = shadowActivity.getNextStartedActivity();

        assertEquals(
                nextIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_TOTAL),
                ORDERS.size());
        assertEquals(
                nextIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_CURRENT),
                1);
        assertEquals(
                nextIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_ORDER_ID),
                ORDERS.get(0).id());
    }

    @Test
    public void processUnique_startsDetails() throws Exception {
        init(ORDERS);

        activity.recyclerView.getChildAt(0).performClick();

        ShadowActivity shadowActivity = shadowOf(activity);

        Intent startedIntent = shadowActivity.getNextStartedActivity();

        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_TOTAL),
                1);
        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_CURRENT),
                1);
        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_ORDER_ID),
                ORDERS.get(0).id());
    }

    @Test
    public void processUnique_resultOkUnique_resetsCurrent() throws Exception {
        init(ORDERS);

        activity.recyclerView.getChildAt(0).performClick();

        ShadowActivity shadowActivity = shadowOf(activity);

        shadowActivity.receiveResult(
                shadowActivity.getNextStartedActivity(),
                OrdersListActivity.RESULT_CODE_OK_UNIQUE,
                null);

        activity.fab.performClick();

        Intent startedIntent = shadowActivity.getNextStartedActivity();

        assertEquals(
```

```
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_TOTAL),
                ORDERS.size());
        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_CURRENT),
                1);
        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_ORDER_ID),
                ORDERS.get(0).id());
    }

    @Test
    public void processUnique_resultClose_resetsCurrent() throws Exception {
        init(ORDERS);

        activity.recyclerView.getChildAt(0).performClick();

        ShadowActivity shadowActivity = shadowOf(activity);

        shadowActivity.receiveResult(
                shadowActivity.getNextStartedActivity(),
                OrdersListActivity.RESULT_CODE_CLOSE,
                null);

        activity.fab.performClick();

        Intent startedIntent = shadowActivity.getNextStartedActivity();

        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_TOTAL),
                ORDERS.size());
        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_CURRENT),
                1);
        assertEquals(
                startedIntent.getExtras().getInt(OrderDetailsActivity.EXTRA_ORDER_ID),
                ORDERS.get(0).id());
    }
}
```

B.3 Código Específico versão MVP

Listagem B.67 – BaseView

```
package br.com.aaascp.gerenciadordepedidos.presentation;

import android.support.annotation.NonNull;

/**
 * Created by andre on 28/09/17.
 */

public interface BaseView<T> {

    void setPresenter(@NonNull T presenter);
}
```

Listagem B.68 – BasePresenter

```
package br.com.aaascp.gerenciadordepedidos.presentation;

/**
 * Created by andre on 28/09/17.
 */

public interface BasePresenter {

    void start();
}
```

Listagem B.69 – BarcodeProcessor

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;

import android.util.SparseArray;

import com.google.android.gms.vision.Detector;
import com.google.android.gms.vision.barcode.Barcode;

import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;

/**
 * Created by andre on 22/09/17.
 */

final class BarcodeProcessor implements Detector.Processor<Barcode> {

    private final BaseActivity baseActivity;
    private final OnItemProcessedListener listener;

    BarcodeProcessor(
            BaseActivity baseActivity,
            OnItemProcessedListener listener) {

        this.baseActivity = baseActivity;
        this.listener = listener;
    }

    @Override
    public void receiveDetections(Detector.Detections<Barcode> detections) {

        final SparseArray<Barcode> barcodes = detections.getDetectedItems();

        if (barcodes.size() != 0) {
            final String code = barcodes.valueAt(0).displayValue;

            baseActivity.runOnUiThread(new Runnable() {
                @Override
                public void run() {
                    listener.onItemProcessed(code);
                }
            });
        }
    }

    @Override
    public void release() {

    }
```

```
}
```

Listagem B.70 – BarcodeProcessorActivity

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;

import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.ColorRes;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v4.content.ContextCompat;
import android.support.v7.widget.Toolbar;
import android.util.ArrayMap;
import android.view.SurfaceView;
import android.view.View;
import android.widget.TextView;

import com.google.android.gms.vision.CameraSource;
import com.google.android.gms.vision.barcode.BarcodeDetector;

import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.DialogUtils;
import br.com.aaascp.gerenciadordepedidos.presentation.util.SnackBarUtils;
import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by andre on 21/09/17.
 */

public final class BarcodeProcessorActivity extends BaseActivity
        implements OnItemProcessedListener, BarcodeProcessorContract.View {


    public static final String EXTRA_RESULT = "EXTRA_RESULT";

    public static final String EXTRA_CODES_TO_PROCESS = "EXTRA_CODES_TO_PROCESS";

    private static final int PROCESSING_TIME = 3 * 1000;

    private BarcodeDetector barcodeDetector;

    @BindView(R.id.barcode_processor_toolbar)
    Toolbar toolbar;

    @BindView(R.id.barcode_processor_preview)
    SurfaceView previewLayout;

    @BindView(R.id.barcode_processor_root)
    View root;

    @BindView(R.id.barcode_processor_guide)
    View guide;
```

```
@BindView(R.id.barcode_processor_title)
TextView title;

@BindView(R.id.barcode_processor_items_left)
TextView itemsLeft;

private BarcodeProcessorContract.Presenter presenter;

public static Intent getIntentForOrder(
        Context context,
        CodesToProcess codesToProcess) {

    Intent intent = new Intent(
            context,
            BarcodeProcessorActivity.class);

    intent.putExtra(EXTRA_CODES_TO_PROCESS, codesToProcess);

    return intent;
}

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_barcode_processor);

    ButterKnife.bind(this);

    new BarcodeProcessorPresenter(
            this,
            getCodesToProcessExtra());

    setupBarcodeDetector();
    setupCamera();
}

@Override
protected void onStart() {
    super.onStart();

    presenter.start();
}

@Override
public void onItemProcessed(String code) {
    presenter.onItemProcessed(code);
}

@Override
public void setupToolbar(int orderId) {
    title.setText(
            String.format(
                    getString(R.string.barcode_processor_title),
                    orderId));

    toolbar.setNavigationIcon(R.drawable.ic_back_white_vector);
    toolbar.setNavigationOnClickListener(new View.OnClickListener() {
        @Override
```

```
        public void onClick(View view) {
            presenter.onFinish();
        }
    });
}

@Override
public void onBackPressed() {
    presenter.onFinish();
}

@Override
public void setItemsLeft(int itemsLeftCount) {
    String itemsLeftText =
            getResources()
                    .getQuantityString(
                            R.plurals.barcode_processor_items_left,
                            itemsLeftCount);

    itemsLeft.setText(
            String.format(
                    itemsLeftText,
                    itemsLeftCount));
}

@Override
public void setZeroItemsLeft() {
    itemsLeft.setText(
            getString(R.string.barcode_processor_items_left));
}

@Override
public void showProcessError() {
    showProcessing(R.color.red);
}

@Override
public void showProcessSuccess() {
    showProcessing(R.color.green);
}

@Override
public void showSuccessMessage(String code) {
    showMessage(String.format(
            getString(R.string.barcode_processor_success_message),
            code));
}

@Override
public void showCodeAlreadyProcessedMessage(String code) {
    showMessage(String.format(
            getString(R.string.barcode_processor_already_processed_message),
            code));
}

@Override
public void showCodeInvalidMessage(String code) {
    showMessage(String.format(
            getString(R.string.barcode_processor_invalid_message),
            code));
```

```
}

@Override
public void showFinishDialog() {
    DialogUtils.showGenericDialog(
            this,
            R.string.barcode_processor_finish_dialog_title,
            R.string.barcode_processor_finish_dialog_message,
            new AlertDialog.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int id) {
                    presenter.onFinish();
                }
            });
}

@Override
public void close(CodesToProcess codesToProcess) {
    Intent result = new Intent();
    result.putExtra(EXTRA_RESULT, codesToProcess);
    setResult(RESULT_OK, result);

    finish();
}

@Override
public void setPresenter(@NonNull BarcodeProcessorContract.Presenter presenter) {
    this.presenter = presenter;
}

private CodesToProcess getCodesToProcessExtra() {
    Bundle extras = getIntent().getExtras();
    if (extras != null) {
        return extras.getParcelable(EXTRA_CODES_TO_PROCESS);
    }

    return CodesToProcess.create(new ArrayMap<String, Integer>(0), Order.INVALID_ORDER_ID);
}

private void showProcessing(final @ColorRes int color) {
    final int normalColor =
            ContextCompat.getColor(
                    this,
                    android.R.color.white);

    final int highlightColor =
            ContextCompat.getColor(
                    this,
                    color);

    guide.getBackground().setTint(highlightColor);

    root.postDelayed(new Runnable() {
        @Override
        public void run() {
            guide.getBackground().setTint(normalColor);
            presenter.onProcessingDone();
        }
    }, PROCESSING_TIME);
}
```

```
    private void setupCamera() {
        CameraSource cameraSource = new CameraSource
                .Builder(this, barcodeDetector)
                .setAutoFocusEnabled(true)
                .build();

        previewLayout.getHolder()
                .addCallback(
                        new CameraPreview(cameraSource, this));
    }

    private void setupBarcodeDetector() {
        barcodeDetector =
                new BarcodeDetector.Builder(this)
                        .build();

        if (!barcodeDetector.isOperational()) {
            return;
        }

        barcodeDetector.setProcessor(
                new BarcodeProcessor(this, this));
    }

    void showMessage(String message) {
        SnackBarUtils.showWithCenteredText(
                root,
                message);
    }
}
```

Listagem B.71 – BarcodeProcessorContract

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;

import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;
import br.com.aaascp.gerenciadordepedidos.presentation.BasePresenter;
import br.com.aaascp.gerenciadordepedidos.presentation.BaseView;

/**
 * Created by andre on 29/09/17.
 */

interface BarcodeProcessorContract {

    interface View extends BaseView<Presenter> {
        void setupToolbar(int orderId);

        void setItemsLeft(int itemsLeftCount);

        void setZeroItemsLeft();


        void showProcessError();

        void showProcessSuccess();

        void showSuccessMessage(String code);
```

```
        void showCodeAlreadyProcessedMessage(String code);

        void showCodeInvalidMessage(String code);

        void showFinishDialog();

        void close(CodesToProcess codesProcessed);
    }

    interface Presenter extends BasePresenter {
        void onItemProcessed(String code);

        void onProcessingDone();

        void onFinish();
    }
}
```

Listagem B.72 – BarcodeProcessorPresenter

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;

import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;

/**
 * Created by andre on 29/09/17.
 */

final class BarcodeProcessorPresenter implements BarcodeProcessorContract.Presenter {

    private final BarcodeProcessorContract.View view;
    private final CodesToProcess codesToProcess;

    private boolean ready;

    BarcodeProcessorPresenter(
            BarcodeProcessorContract.View view,
            CodesToProcess codesToProcess) {

        this.view = view;
        this.codesToProcess = codesToProcess;

        ready = true;
        view.setPresenter(this);
    }

    @Override
    public void start() {
        setup();
    }

    private void setup() {
        view.setupToolbar(codesToProcess.orderId());
        setItemsLeft();
    }

    private void setItemsLeft() {
        int itemsLeft = codesToProcess.itemsLeft();

        if (codesToProcess.itemsLeft() > 0) {
```

```
            view.setItemsLeft(itemsLeft);
        } else {
            view.setZeroItemsLeft();
        }
    }

    @Override
    public void onItemProcessed(String code) {
        if (!ready) {
            return;
        }
        ready = false;

        CodesToProcess.Status status = codesToProcess.process(code);

        if (status == CodesToProcess.Status.SUCCESS) {
            onProcessSuccess();
        } else {
            view.showProcessError();
        }

        showProcessMessage(status, code);
    }

    @Override
    public void onProcessingDone() {
        ready = true;
    }

    @Override
    public void onFinish() {
        view.close(codesToProcess);
    }

    private void onProcessSuccess() {
        view.showProcessSuccess();
        setItemsLeft();
        checkFinish();
    }

    private void showProcessMessage(CodesToProcess.Status status, String code) {
        switch (status) {
            case SUCCESS:
                view.showSuccessMessage(code);
                break;
            case CODE_ALREADY_PROCESSED:
                view.showCodeAlreadyProcessedMessage(code);
                break;
            case CODE_INVALID:
                view.showCodeInvalidMessage(code);
                break;
        }
    }

    private void checkFinish() {
        if (codesToProcess.itemsLeft() == 0) {
            view.showFinishDialog();
        }
    }
}
```

Listagem B.73 – CameraPreview

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;

import android.util.Log;
import android.view.SurfaceHolder;

import com.google.android.gms.vision.CameraSource;

import java.io.IOException;

import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import br.com.aaascp.gerenciadordepedidos.util.PermissionUtils;

/**
 * Created by andre on 21/09/17.
 */

final class CameraPreview implements SurfaceHolder.Callback {

    private static final String TAG = CameraPreview.class.getSimpleName();

    private final CameraSource cameraSource;
    private final BaseActivity activity;

    CameraPreview(CameraSource cameraSource, BaseActivity activity) {
        this.cameraSource = cameraSource;
        this.activity = activity;
    }

    @Override
    public void surfaceCreated(SurfaceHolder holder) {
        try {
            if(PermissionUtils.isCameraEnabled(activity)) {
                cameraSource.start(holder);
            } else {
                activity.finish();
            }
        } catch (IOException e) {
            Log.e(TAG, e.getMessage());
        }
    }

    @Override
    public void surfaceChanged(
            SurfaceHolder holder,
            int format,
            int width,
            int height) {

        //
    }

    @Override
    public void surfaceDestroyed(SurfaceHolder holder) {
        cameraSource.stop();
    }
}
```

Listagem B.74 – DashboardContract

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.main.dashboard;

import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.presentation.BasePresenter;
import br.com.aaascp.gerenciadordepedidos.presentation.BaseView;
import br.com.aaascp.gerenciadordepedidos.presentation.util.DialogUtils;

/**
 * Created by andre on 28/09/17.
 */

interface DashboardContract {
    interface View extends BaseView<Presenter> {
        void setToProcessCount(String count);

        void showGetIdsDialog(DialogUtils.IntValuesListener listener);

        void showErrorGettingIds();

        void navigateToOrdersList(OrderFilterList filter, boolean showProcessAll);
    }

    interface Presenter extends BasePresenter {

        void onToProcessButtonClicked();

        void onProcessedButtonClicked();

        void onAllButtonClicked();

        void onFindButtonClicked();
    }
}
```

Listagem B.75 – DashboardFragment

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.main.dashboard;

import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;

import br.com.aaascp.gerenciadordepedidos.Inject;
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.presentation.custom.ValueLabelView;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseFragment;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list.OrdersListActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.DialogUtils;
import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

/**
 * Created by andre on 18/09/17.
 */
```

```
public final class DashboardFragment extends BaseFragment implements DashboardContract.View {

    @BindView(R.id.dashboard_to_process)
    ValueLabelView toProcessButton;

    private DashboardContract.Presenter presenter;

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        new DashboardPresenter(this, Inject.provideOrdersRepository());
    }

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle save
        super.onCreateView(inflater, container, savedInstanceState);

        View view = inflater.inflate(
                R.layout.fragment_dashboard,
                container,
                false);

        ButterKnife.bind(this, view);

        return view;
    }

    @Override
    public void onStart() {
        super.onStart();

        presenter.start();
    }

    @Override
    public void setToProcessCount(String count) {
        toProcessButton.setValue(count);
    }

    @Override
    public void navigateToOrdersList(OrderFilterList filters, boolean processAll) {
        OrdersListActivity.startForContext(
                getActivity(),
                filters,
                processAll);
    }

    @Override
    public void showGetIdsDialog(DialogUtils.IntValuesListener listener) {
        DialogUtils.showGetIntValues(
                getActivity(),
                R.string.dashboard_orders_find_dialog_title,
                R.string.dashboard_orders_find_dialog_message,
                listener);
    }

    @Override
    public void showErrorGettingIds() {
```

```
            DialogUtils.showError(
                    getActivity(),
                    getString(R.string.dashboard_orders_find_dialog_error_title),
                    getString(R.string.dashboard_orders_find_dialog_erro_message));
        }

        @OnClick(R.id.dashboard_to_process)
        void toProcessButton() {
            presenter.onToProcessButtonClicked();
        }

        @OnClick(R.id.dashboard_processed)
        void processedButton() {
            presenter.onProcessedButtonClicked();
        }

        @OnClick(R.id.dashboard_all)
        void allButton() {
            presenter.onAllButtonClicked();
        }

        @OnClick(R.id.dashboard_find)
        void findButton() {
            presenter.onFindButtonClicked();
        }

        @Override
        public void setPresenter(@NonNull DashboardContract.Presenter presenter) {
            this.presenter = presenter;
        }
}
```

Listagem B.76 – DashboardPresenter

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.main.dashboard;

import java.util.ArrayList;
import java.util.HashSet;
import java.util.List;

import br.com.aaascp.gerenciadordepedidos.Inject;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.presentation.util.DialogUtils;
import br.com.aaascp.gerenciadordepedidos.repository.OrdersRepository;
import br.com.aaascp.gerenciadordepedidos.repository.callback.RepositoryCallback;
import br.com.aaascp.gerenciadordepedidos.repository.filter.IdFilter;
import br.com.aaascp.gerenciadordepedidos.repository.filter.OrderFilter;
import br.com.aaascp.gerenciadordepedidos.repository.filter.StatusFilter;

/**
 * Created by andre on 28/09/17.
 */

final class DashboardPresenter implements DashboardContract.Presenter {

    private final DashboardContract.View view;
    private final OrdersRepository ordersRepository;

    DashboardPresenter(
```

```
            DashboardContract.View view,
            OrdersRepository ordersRepository) {

        this.view = view;
        this.ordersRepository = ordersRepository;

        view.setPresenter(this);
    }

    @Override
    public void start() {
        setupToProcessButton();
    }

    private void setupToProcessButton() {
        StatusFilter filter = StatusFilter.create(StatusFilter.Status.TO_PROCESS);

        ordersRepository.getList(
                filter,
                new RepositoryCallback<List<Order>>() {
                    @Override
                    public void onSuccess(List<Order> data) {
                        view.setToProcessCount(
                                String.valueOf(data.size()));
                    }

                    @Override
                    public void onError(List<String> errors) {
                        view.setToProcessCount("?");
                    }
                });

    }

    @Override
    public void onToProcessButtonClicked() {
        navigateToOrdersList(
                StatusFilter.create(StatusFilter.Status.TO_PROCESS),
                true);
    }

    @Override
    public void onProcessedButtonClicked() {
        navigateToOrdersList(
                StatusFilter.create(StatusFilter.Status.PROCESSED),
                false);
    }

    @Override
    public void onAllButtonClicked() {
        navigateToOrdersList(
                StatusFilter.create(StatusFilter.Status.ALL),
                false);
    }

    @Override
    public void onFindButtonClicked() {
        view.showGetIdsDialog(new DialogUtils.IntValuesListener() {
            @Override
            public void onValues(HashSet<Integer> values) {
```

```
                navigateToOrdersList(
                        IdFilter.create(values),
                        values.size() > 1);
            }

            @Override
            public void onError() {
                view.showErrorGettingIds();
            }
        });
    }

    private void navigateToOrdersList(OrderFilter filter, boolean showProcessAll) {
        List<OrderFilter> filters = new ArrayList<>();
        filters.add(filter);

        view.navigateToOrdersList(
                OrderFilterList.create(filters),
                showProcessAll);
    }
}
```

Listagem B.77 – MainActivity

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.main;

import android.os.Bundle;

import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import butterknife.ButterKnife;

public final class MainActivity extends BaseActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_main);
        ButterKnife.bind(this);
    }
}
```

Listagem B.78 – OnItemProcessedListener

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;

/**
 * Created by andre on 22/09/17.
 */

interface OnItemProcessedListener {

    void onItemProcessed(String code);
}
```

Listagem B.79 – OrderDetailsAdapter

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.details;
```

```
import android.app.AlertDialog;
import android.content.Context;
import android.content.DialogInterface;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.TextView;

import br.com.aaascp.gerenciadordepedidos.Inject;
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.presentation.custom.ValueLabelView;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.camera.BarcodeProcessorActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.info.OrderInfoActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list.OrdersListActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.DialogUtils;
import br.com.aaascp.gerenciadordepedidos.presentation.util.EmptyStateAdapter;
import br.com.aaascp.gerenciadordepedidos.util.PermissionUtils;
import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

/**
 * Created by andre on 09/07/17.
 */

public class OrderDetailsActivity extends BaseActivity implements OrderDetailsContract.View {

    private static final int REQUEST_CODE_CAMERA_PERMISSION = 100;
    static final int REQUEST_CODE_PROCESS = 200;

    public static final String EXTRA_ORDER_ID = "EXTRA_ORDER_ID";
    public static final String EXTRA_TOTAL = "EXTRA_TOTAL";
    public static final String EXTRA_CURRENT = "EXTRA_CURRENT";

    @BindView(R.id.order_details_toolbar)
    Toolbar toolbar;

    @BindView(R.id.order_details_recycler)
    RecyclerView recyclerView;

    @BindView(R.id.order_details_items_left)
    TextView itemsLeftView;

    @BindView(R.id.order_details_ship_type)
    ValueLabelView shipTypeView;

    @BindView(R.id.order_details_finish_root)
    View finishRoot;

    @BindView(R.id.order_details_processed_root)
    View alreadyProcessedRoot;
```

```
@BindView(R.id.order_details_items_left_root)
View itemsLeftRoot;

OrderDetailsContract.Presenter presenter;
private OrderDetailsAdapter orderDetailsAdapter;

public static Intent getIntentForOrder(
        Context context,
        int orderId,
        int current,
        int total) {

    Intent intent = new Intent(
            context,
            OrderDetailsActivity.class);

    intent.putExtra(EXTRA_ORDER_ID, orderId);
    intent.putExtra(EXTRA_TOTAL, total);
    intent.putExtra(EXTRA_CURRENT, current);

    return intent;
}

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    setContentView(R.layout.activity_order_details);
    ButterKnife.bind(this);

    Bundle extras = getIntent().getExtras();

    if (extras != null) {
        new OrderDetailsPresenter(
                this,
                Inject.provideOrdersRepository(),
                extras.getInt(EXTRA_ORDER_ID, Order.INVALID_ORDER_ID),
                extras.getInt(EXTRA_TOTAL, 1),
                extras.getInt(EXTRA_CURRENT, 1));
    } else {
        new OrderDetailsPresenter(
                this,
                Inject.provideOrdersRepository(),
                Order.INVALID_ORDER_ID,
                1,
                1);
    }
}

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (data == null) {
        return;
    }

    Bundle extras = data.getExtras();
    if (resultCode == RESULT_OK &&
```

```
            requestCode == REQUEST_CODE_PROCESS &&
            extras != null) {

        CodesToProcess codesToProcess =
                extras.getParcelable(
                        BarcodeProcessorActivity.EXTRA_RESULT);

        presenter.onProcessorResult(codesToProcess);
    }
}

@Override
protected void onStart() {
    super.onStart();

    presenter.start();
}

@Override
public void setupMenu() {
    toolbar.inflateMenu(R.menu.menu_order_details);

    toolbar.setOnMenuItemClickListener(new Toolbar.OnMenuItemClickListener() {
        @Override
        public boolean onMenuItemClick(MenuItem item) {
            int id = item.getItemId();

            if (id == R.id.menu_order_details_more_details) {
                presenter.onInfoClicked();
                return true;
            } else if (id == R.id.menu_order_details_clear) {
                presenter.onClearClicked();
                return true;
            } else if (id == R.id.menu_order_details_skip) {
                presenter.onSkipClicked();
                return true;
            }
            return false;
        }
    });

    presenter.onMenuCreated();
}

@Override
public void hideClearButton() {
    toolbar.getMenu()
            .findItem(R.id.menu_order_details_clear).setVisible(false);
}

@Override
public void hideSkipButton() {
    toolbar.getMenu()
            .findItem(R.id.menu_order_details_skip).setVisible(false);
}

@Override
public void updateCodesProcessed(CodesToProcess codesToProcess) {
    orderDetailsAdapter.updateCodesProcessed(codesToProcess);
}
```

```
@Override
public void setupCloseToolbar() {
    toolbar.setNavigationIcon(R.drawable.ic_close_white_vector);
    toolbar.setNavigationOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    presenter.onCloseClicked();
                }
            });
}

@Override
public void setupBackToolbar() {
    toolbar.setNavigationIcon(R.drawable.ic_back_white_vector);
    toolbar.setNavigationOnClickListener(
            new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    presenter.onBackPressed();
                }
            });
}

@Override
public void setupTitle(int id, int current, int total) {
    String title =
            getResources().getQuantityString(
                    R.plurals.order_details_title,
                    total);

    String titleFormatted =
            String.format(
                    title,
                    id,
                    current,
                    total);

    toolbar.setTitle(titleFormatted);
}

@Override
public void setShipType(String shipType) {
    shipTypeView.setValue(shipType);
}

@Override
public void setItemsLeft(int total, int itemsLeftCount) {
    itemsLeftView.setText(
            String.format(
                    getString(R.string.order_details_count_text),
                    total - itemsLeftCount,
                    total));
}

@Override
public void showOrder(Order order, CodesToProcess codesToProcess) {
    orderDetailsAdapter =
            new OrderDetailsAdapter(
```

```
                    this,
                    order.items(),
                    codesToProcess);

    recyclerView.setAdapter(orderDetailsAdapter);
}

@Override
public void showError(String error) {
    recyclerView.setAdapter(
            new EmptyStateAdapter(
                    this,
                    error));
}

@Override
public void showCommunicationError() {
    showError(
            getString(R.string.error_communication));
}

@Override
public void hideLabels() {
    finishRoot.setVisibility(View.GONE);
    itemsLeftRoot.setVisibility(View.GONE);
    alreadyProcessedRoot.setVisibility(View.GONE);
}

@Override
public void showFinishLabel() {
    finishRoot.setVisibility(View.VISIBLE);
}

@Override
public void showItemsLeftLabel() {
    itemsLeftRoot.setVisibility(View.VISIBLE);
}

@Override
public void showAlreadyProcessedLabel() {
    alreadyProcessedRoot.setVisibility(View.VISIBLE);
}

@Override
public void navigateToDetails(Order order) {
    OrderInfoActivity.startForOrder(this, order);
}

@Override
public void finishSkip() {
    finishOrder(OrdersListActivity.RESULT_CODE_SKIP);
}

@Override
public void finishClose() {
    finishOrder(OrdersListActivity.RESULT_CODE_CLOSE);
}

@Override
public void finishOk() {
```

```
        finishOrder(OrdersListActivity.RESULT_CODE_OK);
    }

    @Override
    public void finishOkUnique() {
        finishOrder(OrdersListActivity.RESULT_CODE_OK_UNIQUE);
    }


    @Override
    public void onBackPressed() {
        presenter.onBackPressed();
    }

    @Override
    public void finishBack() {
        super.onBackPressed();
    }

    @Override
    public void showBackDialog() {
        DialogUtils.showGenericDialog(
                this,
                R.string.order_details_back_dialog_title,
                R.string.order_details_back_dialog_message,
                new AlertDialog.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int id) {
                        presenter.onBackDialogOk();
                    }
                }
        );
    }

    @Override
    public void showSkipDialog() {
        DialogUtils.showGenericDialog(
                this,
                R.string.order_details_skip_dialog_title,
                R.string.order_details_skip_dialog_message,
                new AlertDialog.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int id) {
                        presenter.onSkipDialogOk();
                    }
                }
        );
    }

    @Override
    public void showClearDialog() {
        DialogUtils.showGenericDialog(
                this,
                R.string.order_details_clear_dialog_title,
                R.string.order_details_clear_dialog_message,
                new AlertDialog.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int id) {
                        presenter.onClearDialogOk();
                    }
```

```
            }
        );
    }

    @Override
    public void showCloseDialog() {
        DialogUtils.showGenericDialog(
                this,
                R.string.order_details_close_dialog_title,
                R.string.order_details_close_dialog_message,
                new AlertDialog.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog, int id) {
                        presenter.onCloseDialogOk();
                    }
                }
        );
    }

    @Override
    public void navigateToProcessor(CodesToProcess codesToProcess) {
        startActivityForResult(
                BarcodeProcessorActivity.getIntentForOrder(
                        this,
                        codesToProcess),
                REQUEST_CODE_PROCESS);
    }

    @OnClick(R.id.order_details_fab)
    void onFabClick() {
        presenter.onFabClicked();
    }

    @OnClick(R.id.order_details_finish)
    void onFinishedClick() {
        presenter.onFinishClicked();
    }

    @OnClick(R.id.order_details_processed)
    void onAlreadyProcessedClick() {
        presenter.onAlreadyProcessedClicked();
    }

    @OnClick(R.id.order_details_ship_type)
    void onShipTypeClick() {
        presenter.onShipTypeClicked();
    }

    @Override
    public void setPresenter(@NonNull OrderDetailsContract.Presenter presenter) {
        this.presenter = presenter;
    }

    @Override
    public void checkPermissionForCamera() {
        if (PermissionUtils.isCameraEnabled(this)) {
            presenter.onCameraPermissionEnabled();
        } else {
            PermissionUtils.requestPermissionForCamera(this, REQUEST_CODE_CAMERA_PERMISSION);
        }
```

```
    }

    private void finishOrder(int resultCode) {
        Intent intent = new Intent();
        setResult(resultCode, intent);

        finish();
    }
}
```

Listagem B.80 – OrderDetailsAdapter

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.details;

import android.content.Context;
import android.support.v4.content.ContextCompat;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageView;
import android.widget.TextView;

import java.util.ArrayList;
import java.util.List;
import java.util.Map;

import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;
import br.com.aaascp.gerenciadordepedidos.entity.OrderItem;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.item.OrderItemActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.ImageLoader;
import br.com.aaascp.gerenciadordepedidos.util.StringUtils;
import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by andre on 20/09/17.
 */

class OrderDetailsAdapter extends RecyclerView.Adapter<OrderDetailsAdapter.ViewHolder> {

    private final Context context;
    private final Map<String, OrderItem> items;
    private final List<String> index;
    private final LayoutInflater layoutInflater;

    private CodesToProcess codesProcessed;

    OrderDetailsAdapter(
            Context context,
            Map<String, OrderItem> items,
            CodesToProcess codesProcessed) {

        this.context = context;
        this.items = items;
        this.codesProcessed = codesProcessed;

        index = new ArrayList<>();
        index.addAll(items.keySet());
```

```
        layoutInflater = LayoutInflater.from(context);
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        return new ViewHolder(
                layoutInflater.inflate(
                        R.layout.row_order_items,
                        parent,
                        false));
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, int position) {
        String code = index.get(position);
        final OrderItem item = items.get(code);

        int quantity = item.quantity();
        int itemsLeft = quantity - codesProcessed.codes().get(item.code());

        String imageUrl = item.imageUrl();
        if (!StringUtils.isNullOrEmpty(imageUrl)) {
            ImageLoader.loadImage(
                    context,
                    imageUrl,
                    holder.image);
        }

        holder.code.setText(
                String.valueOf(item.code()));

        holder.quantity.setTextColor(
                ContextCompat.getColor(
                        context,
                        itemsLeft == quantity ? R.color.green : R.color.red));

        holder.quantity.setText(
                String.format(
                        context.getString(R.string.order_details_count_text),
                        itemsLeft,
                        quantity));

        holder.description.setText(item.description());

        holder.root.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                OrderItemActivity.startForItem(context, item);
            }
        });
    }

    @Override
    public int getItemCount() {
        return items.size();
    }

    public void updateCodesProcessed(CodesToProcess codesProcessed) {
        this.codesProcessed = codesProcessed;
```

```
        notifyDataSetChanged();
    }

    static class ViewHolder extends RecyclerView.ViewHolder {

        @BindView(R.id.order_item_root)
        View root;

        @BindView(R.id.order_item_image)
        ImageView image;

        @BindView(R.id.order_item_code_value)
        TextView code;

        @BindView(R.id.order_item_quantity_value)
        TextView quantity;

        @BindView(R.id.order_item_description_value)
        TextView description;


        ViewHolder(View itemView) {
            super(itemView);

            ButterKnife.bind(this, itemView);
        }
    }
}
```

Listagem B.81 – OrderDetailsContract

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.details;

import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.presentation.BasePresenter;
import br.com.aaascp.gerenciadordepedidos.presentation.BaseView;

/**
 * Created by andre on 30/09/17.
 */

interface OrderDetailsContract {

    interface View extends BaseView<Presenter> {
        void hideClearButton();

        void hideSkipButton();

        void setupMenu();

        void setupCloseToolbar();

        void setupBackToolbar();

        void setupTitle(int id, int current, int total);

        void setShipType(String shipType);

        void setItemsLeft(int total, int itemsLeft);
```

```
        void showOrder(Order order, CodesToProcess codesToProcess);

        void showError(String error);

        void showCommunicationError();

        void finishOkUnique();

        void showBackDialog();

        void showSkipDialog();

        void showClearDialog();

        void showCloseDialog();

        void hideLabels();

        void showFinishLabel();

        void showItemsLeftLabel();

        void showAlreadyProcessedLabel();

        void finishBack();

        void finishSkip();

        void finishClose();

        void finishOk();

        void updateCodesProcessed(CodesToProcess codesToProcess);

        void navigateToDetails(Order order);

        void navigateToProcessor(CodesToProcess codesToProcess);

        void checkPermissionForCamera();
    }

    interface Presenter extends BasePresenter {

        void onMenuCreated();

        void onBackPressed();

        void onInfoClicked();

        void onClearClicked();

        void onSkipClicked();

        void onCloseClicked();

        void onBackDialogOk();

        void onSkipDialogOk();
```

```
        void onClearDialogOk();

        void onCloseDialogOk();

        void onFabClicked();

        void onFinishClicked();

        void onAlreadyProcessedClicked();

        void onShipTypeClicked();

        void onProcessorResult(CodesToProcess codesToProcess);

        void onCameraPermissionEnabled();
    }
}
```

Listagem B.82 – OrderDetailsPresenter

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.details;

import java.util.List;

import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.repository.OrdersRepository;
import br.com.aaascp.gerenciadordepedidos.repository.callback.RepositoryCallback;
import br.com.aaascp.gerenciadordepedidos.util.DateFormatterUtils;
import br.com.aaascp.gerenciadordepedidos.util.PermissionUtils;

/**
 * Created by andre on 30/09/17.
 */

final class OrderDetailsPresenter implements OrderDetailsContract.Presenter {

    private final OrderDetailsContract.View view;

    private final OrdersRepository ordersRepository;

    private final int orderId;
    private final int total;
    private final int current;

    private CodesToProcess codesToProcess;
    private Order order;

    OrderDetailsPresenter(
            OrderDetailsContract.View view,
            OrdersRepository ordersRepository,
            int orderId,
            int total,
            int current) {

        this.view = view;
        this.ordersRepository = ordersRepository;

        this.orderId = orderId;
        this.total = total;
```

```
        this.current = current;


        view.setPresenter(this);
        setupToolbar();
    }

    @Override
    public void start() {
        setupOrder();
    }

    @Override
    public void onMenuCreated() {
        if (total == current) {
            view.hideSkipButton();
        }
    }

    @Override
    public void onBackPressed() {
        if (order.isProcessed()) {
            view.finishBack();
        } else {
            view.showBackDialog();
        }
    }

    @Override
    public void onInfoClicked() {
        if (order != null) {
            view.navigateToDetails(order);
        }
    }

    @Override
    public void onProcessorResult(CodesToProcess codesToProcess) {
        this.codesToProcess = codesToProcess;
        view.updateCodesProcessed(codesToProcess);
        checkFinish();
    }

    @Override
    public void onClearClicked() {
        view.showClearDialog();
    }

    @Override
    public void onSkipClicked() {
        view.showSkipDialog();
    }

    @Override
    public void onCloseClicked() {
        view.showCloseDialog();
    }

    @Override
    public void onBackDialogOk() {
        view.finishBack();
```

```
}

@Override
public void onSkipDialogOk() {
    view.finishSkip();
}

@Override
public void onClearDialogOk() {
    order = null;
    setupOrder();
}

@Override
public void onCloseDialogOk() {
    view.finishClose();
}

@Override
public void onFabClicked() {
    view.checkPermissionForCamera();
}

@Override
public void onCameraPermissionEnabled() {
    view.navigateToProcessor(codesToProcess);
}

@Override
public void onFinishClicked() {
    ordersRepository.save(
            order.withProcessedAt(
                    DateFormatterUtils
                            .getDateHourInstance()
                            .now()));

    if (total == 1) {
        view.finishOkUnique();
    } else {
        view.finishOk();
    }
}

@Override
public void onAlreadyProcessedClicked() {
    view.finishClose();
}

@Override
public void onShipTypeClicked() {
    view.navigateToDetails(order);
}

private void setupOrder() {
    if (order != null) {
        view.showOrder(order, codesToProcess);
        return;
    }

    ordersRepository.getOrder(
```

```
                orderId,
                new RepositoryCallback<Order>() {
                    @Override
                    public void onSuccess(Order data) {
                        order = data;
                        codesToProcess = order.codesToProcess();

                        showOrder();
                    }

                    @Override
                    public void onError(List<String> errors) {
                        if (errors != null) {
                            view.showError(errors.get(0));
                        } else {
                            view.showCommunicationError();
                        }
                    }
                }
        );
    }

    private void showOrder() {
        checkFinish();

        view.setShipType(order.shipmentInfo().shipType());
        view.showOrder(order, codesToProcess);
    }

    void checkFinish() {
        int itemsLeft = codesToProcess.itemsLeft();

        view.hideLabels();

        if (order.isProcessed()) {
            view.hideClearButton();
            view.showAlreadyProcessedLabel();
        } else if (itemsLeft == 0) {
            view.showFinishLabel();
        } else {
            view.showItemsLeftLabel();
            view.setItemsLeft(order.size(), itemsLeft);
        }
    }

    private void setupToolbar() {
        view.setupMenu();
        view.setupTitle(orderId, current, total);

        if (total > 1) {
            view.setupCloseToolbar();
        } else {
            view.setupBackToolbar();
        }
    }
}
```

Listagem B.83 – OrderInfoActivity

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.info;
```

```
import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.TextView;

import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by andre on 27/09/17.
 */

public final class OrderInfoActivity extends BaseActivity implements OrderInfoContract.View {

    public static final String EXTRA_ORDER = "EXTRA_ORDER";

    @BindView(R.id.order_info_toolbar)
    Toolbar toolbar;

    @BindView(R.id.info_order_size_value)
    TextView orderSize;

    @BindView(R.id.info_order_processed_at_value)
    TextView orderProcessedAt;

    @BindView(R.id.info_order_last_modification_value)
    TextView orderLastModification;

    @BindView(R.id.info_shipment_type_value)
    TextView shipmentType;

    @BindView(R.id.info_shipment_address_value)
    TextView shipmentAddress;

    @BindView(R.id.info_customer_id_value)
    TextView customerId;

    @BindView(R.id.info_customer_name_value)
    TextView customerName;

    private OrderInfoContract.Presenter presenter;

    public static void startForOrder(Context context, Order order) {
        Intent intent = new Intent(context, OrderInfoActivity.class);

        intent.putExtra(EXTRA_ORDER, order);

        context.startActivity(intent);
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_order_info);
        ButterKnife.bind(this);

        new OrderInfoPresenter(this, getOrderExtra());
    }

    private Order getOrderExtra() {
        Bundle extras = getIntent().getExtras();

        if (extras != null) {
            return extras.getParcelable(EXTRA_ORDER);
        }

        return null;
    }

    @Override
    protected void onStart() {
        super.onStart();

        presenter.start();
    }

    @Override
    public void setupToolbar(int id) {
        toolbar.setTitle(
                String.format(
                        getString(R.string.info_title),
                        id));

        toolbar.setNavigationIcon(R.drawable.ic_back_white_vector);

        toolbar.setNavigationOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                onBackPressed();
            }
        });
    }

    @Override
    public void setProcessedOrderInfo(String size, String processedAt, String lastModification) {
        setOrderInfo(size, processedAt, lastModification);
    }

    @Override
    public void setNotProcessedOrderInfo(String size, String lastModification) {
        setOrderInfo(
                size,
                getString(R.string.info_order_processed_at_empty),
                lastModification);
    }

    @Override
    public void setShipmentInfo(String type, String address) {
        shipmentType.setText(type);
        shipmentAddress.setText(address);
    }
```

```
    @Override
    public void setCustomerInfo(String id, String name) {
        customerId.setText(id);
        customerName.setText(name);
    }

    @Override
    public void setPresenter(@NonNull OrderInfoContract.Presenter presenter) {
        this.presenter = presenter;
    }

    private void setOrderInfo(String size, String processedAt, String lastModification) {
        orderSize.setText(size);
        orderProcessedAt.setText(processedAt);
        orderLastModification.setText(lastModification);
    }
}
```

Listagem B.84 – OrderInfoContract

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.info;

import br.com.aaascp.gerenciadordepedidos.presentation.BasePresenter;
import br.com.aaascp.gerenciadordepedidos.presentation.BaseView;

/**
 * Created by andre on 29/09/17.
 */

interface OrderInfoContract {

    interface View extends BaseView<Presenter> {
        void setupToolbar(int id);

        void setProcessedOrderInfo(String size, String processedAt, String lastModification);

        void setNotProcessedOrderInfo(String size, String lastModification);

        void setShipmentInfo(String type, String address);

        void setCustomerInfo(String id, String name);
    }

    interface Presenter extends BasePresenter {

    }
}
```

Listagem B.85 – OrderInfoPresenter

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.info;

import br.com.aaascp.gerenciadordepedidos.entity.Order;

/**
 * Created by andre on 29/09/17.
 */
final class OrderInfoPresenter implements OrderInfoContract.Presenter {
```

```
    private final OrderInfoContract.View view;
    private final Order order;

    OrderInfoPresenter(OrderInfoContract.View view, Order order) {

        this.view = view;
        this.order = order;

        view.setPresenter(this);
    }

    @Override
    public void start() {
        if (order != null) {
            setupInfo();
        }
    }

    private void setupInfo() {
        view.setupToolbar(order.id());
        setOrderInfo();

        view.setCustomerInfo(
                String.valueOf(order.customerInfo().id()),
                order.customerInfo().name());

        view.setShipmentInfo(
                order.shipmentInfo().shipType(),
                order.shipmentInfo().address());
    }

    private void setOrderInfo() {
        String size = String.valueOf(order.size());
        String lastModification = order.lastModifiedAt();

        if (order.isProcessed()) {
            view.setProcessedOrderInfo(
                    size,
                    order.processedAt(),
                    lastModification);
        } else {
            view.setNotProcessedOrderInfo(size, lastModification);
        }
    }
}
```

Listagem B.86 – OrderItemActivity

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.item;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.v7.widget.Toolbar;
import android.view.View;
import android.widget.ImageView;
import android.widget.TextView;
```

```
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.OrderItem;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.ImageLoader;
import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by andre on 27/09/17.
 */

public final class OrderItemActivity extends BaseActivity implements OrderItemContract.View {

    public static final String EXTRA_ITEM = "EXTRA_ITEM";

    @BindView(R.id.order_item_toolbar)
    Toolbar toolbar;

    @BindView(R.id.order_item_image)
    ImageView imageView;

    @BindView(R.id.order_item_description)
    TextView descriptionView;

    private OrderItemContract.Presenter presenter;

    public static void startForItem(Context context, OrderItem item) {
        Intent intent = new Intent(context, OrderItemActivity.class);

        intent.putExtra(EXTRA_ITEM, item);

        context.startActivity(intent);
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_order_item);
        ButterKnife.bind(this);

        new OrderItemPresenter(this, getOrderItemExtra());
    }

    private OrderItem getOrderItemExtra() {
        Bundle extras = getIntent().getExtras();

        if (extras != null) {
            return extras.getParcelable(EXTRA_ITEM);
        }

        return null;
    }

    @Override
    protected void onStart() {
        super.onStart();

        presenter.start();
```

```
    }

    @Override
    public void setupToolbar(String code) {
        toolbar.setNavigationIcon(R.drawable.ic_close_white_vector);
        toolbar.setNavigationOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                finish();
            }
        });

        toolbar.setTitle(
                String.format(
                        getString(R.string.order_item_title),
                        code));
    }

    @Override
    public void loadImage(@NonNull String imageUrl) {
        ImageLoader.loadImage(
                this,
                imageUrl,
                imageView);
    }

    @Override
    public void setDescription(String description) {
        descriptionView.setText(description);
    }

    @Override
    public void setPresenter(@NonNull OrderItemContract.Presenter presenter) {
        this.presenter = presenter;
    }
}
```

Listagem B.87 – OrderItemContract

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.item;

import android.support.annotation.NonNull;

import br.com.aaascp.gerenciadordepedidos.presentation.BasePresenter;
import br.com.aaascp.gerenciadordepedidos.presentation.BaseView;

/**
 * Created by andre on 28/09/17.
 */

interface OrderItemContract {
    interface View extends BaseView<Presenter> {
        void setupToolbar(String code);

        void loadImage(@NonNull String imageUrl);

        void setDescription(String description);
    }

    interface Presenter extends BasePresenter {
```

```
    }

}
```

Listagem B.88 – OrderItemPresenter

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.item;

import br.com.aaascp.gerenciadordepedidos.entity.OrderItem;
import br.com.aaascp.gerenciadordepedidos.util.StringUtils;

/**
 * Created by andre on 28/09/17.
 */

class OrderItemPresenter implements OrderItemContract.Presenter {

    private final OrderItemContract.View view;
    private final OrderItem item;

    OrderItemPresenter(
            OrderItemContract.View view,
            OrderItem item) {

        this.view = view;
        this.item = item;

        view.setPresenter(this);
    }

    @Override
    public void start() {
        if(item != null) {
            setupItem();
        }
    }

    private void setupItem() {
        view.setupToolbar(item.code());
        view.setDescription(item.description());
        setupImage();
    }

    private void setupImage() {
        String imageUrl = item.imageUrl();
        if (!StringUtils.isNullOrEmpty(imageUrl)) {
            view.loadImage(imageUrl);
        }
    }
}
```

Listagem B.89 – OrdersListActivity

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list;

import android.content.Context;
import android.content.Intent;
import android.os.Bundle;
```

```
import android.support.annotation.NonNull;
import android.support.annotation.Nullable;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.widget.RecyclerView;
import android.support.v7.widget.Toolbar;
import android.view.View;

import java.util.List;

import br.com.aaascp.gerenciadordepedidos.Inject;
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.NullOrderFilterList;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.BaseActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.details.OrderDetailsActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.EmptyStateAdapter;
import butterknife.BindView;
import butterknife.ButterKnife;
import butterknife.OnClick;

/**
 * Created by andre on 09/07/17.
 */

public final class OrdersListActivity extends BaseActivity implements OrdersListContract.View {

    public static final int REQUEST_CODE_ORDER_PROCESS = 100;

    public static final int RESULT_CODE_OK_UNIQUE = 200;
    public static final int RESULT_CODE_OK = 300;
    public static final int RESULT_CODE_SKIP = 400;
    public static final int RESULT_CODE_CLOSE = 500;

    public static final String EXTRA_ORDER_FILTERS = "EXTRA_ORDER_FILTERS";
    public static final String EXTRA_PROCESS_ALL = "EXTRA_PROCESS_ALL";

    @BindView(R.id.orders_list_fab)
    FloatingActionButton fab;

    @BindView(R.id.orders_list_toolbar)
    Toolbar toolbar;

    @BindView(R.id.orders_list_recycler)
    RecyclerView recyclerView;

    OrdersListContract.Presenter presenter;

    public static void startForContext(
            Context context,
            OrderFilterList filters,
            boolean processAll) {

        Intent intent =
                new Intent(
                        context,
                        OrdersListActivity.class);

        intent.putExtra(EXTRA_ORDER_FILTERS, filters);
        intent.putExtra(EXTRA_PROCESS_ALL, processAll);
```

```
        context.startActivity(intent);
    }

    @Override
    protected void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.activity_orders_list);
        ButterKnife.bind(this);

        Bundle extras = getIntent().getExtras();
        if(extras  != null) {
            new OrdersListPresenter(
                    this,
                    getOrderFilterListExtra(extras),
                    Inject.provideOrdersRepository(),
                    getProcessAllExtra(extras));
        } else {
            new OrdersListPresenter(
                    this,
                    NullOrderFilterList.create(),
                    Inject.provideOrdersRepository(),
                    false);
        }

        setupToolbar();
    }

    @Override
    protected void onStart() {
        super.onStart();

        presenter.start();
    }

    @Override
    protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);

        if (requestCode == REQUEST_CODE_ORDER_PROCESS) {
            switch (resultCode) {
                case RESULT_CODE_OK:
                case RESULT_CODE_SKIP:
                    presenter.onResultNext();
                    break;
                case RESULT_CODE_OK_UNIQUE:
                case RESULT_CODE_CLOSE:
                default:
                    presenter.onResultClose();
            }
        }
    }

    private OrderFilterList getOrderFilterListExtra(Bundle extras) {
        return extras.getParcelable(EXTRA_ORDER_FILTERS);
    }

    private boolean getProcessAllExtra(Bundle extras) {
        return extras.getBoolean(EXTRA_PROCESS_ALL, false);
```

```
}

void setupToolbar() {
    toolbar.setNavigationIcon(R.drawable.ic_back_white_vector);
    toolbar.setNavigationOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            onBackPressed();
        }
    });
}

@Override
public void showFab() {
    fab.setVisibility(View.VISIBLE);
}

@Override
public void hideFab() {
    fab.setVisibility(View.GONE);
}

@Override
public void showOrdersList(List<Order> orders) {
  recyclerView.setAdapter(
          new OrdersListAdapter(
                  this,
                  orders,
                  new OrdersListAdapter.OnClickListener() {
                      @Override
                      public void onClick(int orderId) {
                          presenter.onOrderClicked(orderId);
                      }
                  }));
}

@Override
public void showEmptyList() {
    recyclerView.setAdapter(
            new EmptyStateAdapter(
                    this,
                    R.drawable.ic_coffee_black_vector,
                    getString(R.string.order_list_empty)));
}

@Override
public void showCommunicationError() {
    showError(
            getString(R.string.error_communication));
}

@Override
public void showError(String error) {
    recyclerView.setAdapter(
            new EmptyStateAdapter(
                    this,
                    error));
}

@Override
```

```
    public void navigateToOrderDetails(int orderId, int position, int total) {
        Intent intent =
                OrderDetailsActivity.getIntentForOrder(
                        this,
                        orderId,
                        position,
                        total);

        startActivityForResult(intent, REQUEST_CODE_ORDER_PROCESS);
    }

    @Override
    public void setPresenter(@NonNull OrdersListContract.Presenter presenter) {
        this.presenter = presenter;
    }

    @OnClick(R.id.orders_list_fab)
    void onFabClick() {
        presenter.onFabCLick();
    }
}
```

Listagem B.90 – OrdersListAdapter

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list;

import android.content.Context;
import android.support.v4.content.ContextCompat;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import java.util.List;

import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import butterknife.BindView;
import butterknife.ButterKnife;

/**
 * Created by andre on 10/07/17.
 */
final class OrdersListAdapter extends RecyclerView.Adapter<OrdersListAdapter.ViewHolder> {

    private final Context context;
    private final List<Order> orders;
    private final LayoutInflater layoutInflater;
    private final OnClickListener listener;

    OrdersListAdapter(
            Context context,
            List<Order> orders,
            OnClickListener listener) {

        this.context = context;
        this.orders = orders;
        this.listener = listener;
```

```
        layoutInflater = LayoutInflater.from(context);
    }

    @Override
    public ViewHolder onCreateViewHolder(ViewGroup parent, int viewType) {
        return new ViewHolder(
                layoutInflater.inflate(
                        R.layout.row_orders_list,
                        parent,
                        false));
    }

    @Override
    public void onBindViewHolder(ViewHolder holder, final int position) {
        final Order order = orders.get(position);
        boolean isProcessed = order.isProcessed();

        holder.id.setText(
                String.valueOf(
                        order.id()));

        holder.shipType.setText(order.shipmentInfo().shipType());

        holder.itemsCount.setText(
                String.valueOf(
                        order.size()));

        holder.lastModifiedAt.setText(order.lastModifiedAt());

        holder.processedAt.setTextColor(
                ContextCompat.getColor(
                        context,
                        isProcessed ? R.color.green : R.color.red));

        holder.processedAt.setText(getProcessedAt(order));

        holder.action.setText(
                context.getString(
                        isProcessed ? R.string.orders_list_action_details : R.string.orders_list_action_pr

        holder.root.setOnClickListener(
                new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        listener.onClick(order.id());
                    }
                });
    }

    @Override
    public int getItemCount() {
        return orders.size();
    }

    private String getProcessedAt(Order order) {
        if (order.isProcessed()) {
            return order.processedAt();
        }

        return context.getString(R.string.order_list_processed_at_empty);
```

```
    }

    static class ViewHolder extends RecyclerView.ViewHolder {

        @BindView(R.id.order_root)
        View root;

        @BindView(R.id.order_id_value)
        TextView id;

        @BindView(R.id.order_ship_type_value)
        TextView shipType;

        @BindView(R.id.order_size_value)
        TextView itemsCount;

        @BindView(R.id.order_processed_at_value)
        TextView processedAt;

        @BindView(R.id.order_last_modification_date_value)
        TextView lastModifiedAt;

        @BindView(R.id.order_action_text)
        TextView action;

        ViewHolder(View itemView) {
            super(itemView);

            ButterKnife.bind(this, itemView);
        }
    }

    interface OnClickListener {
        void onClick(int orderId);
    }
}
```

Listagem B.91 – OrdersListContract

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list;

import java.util.List;

import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.presentation.BasePresenter;
import br.com.aaascp.gerenciadordepedidos.presentation.BaseView;

/**
 * Created by andre on 30/09/17.
 */

interface OrdersListContract {
    interface View extends BaseView<Presenter> {
        void showOrdersList(List<Order> orders);

        void showEmptyList();

        void showError(String error);

        void showCommunicationError();
```

```
        void showFab();

        void hideFab();

        void navigateToOrderDetails(int orderId, int position, int total);
    }

    interface Presenter extends BasePresenter {
        void onResultNext();

        void onResultClose();

        void onOrderClicked(int orderId);

        void onFabCLick();

    }
}
```

Listagem B.92 – OrdersListPresenter

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list;

import java.util.List;

import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.repository.OrdersRepository;
import br.com.aaascp.gerenciadordepedidos.repository.callback.RepositoryCallback;

/**
 * Created by andre on 30/09/17.
 */

final class OrdersListPresenter implements OrdersListContract.Presenter {

    private final OrdersListContract.View view;
    private final OrdersRepository ordersRepository;
    private final OrderFilterList filters;
    private final boolean processAll;

    private List<Order> orders;
    private int current;

    OrdersListPresenter(
            OrdersListContract.View view,
            OrderFilterList filters,
            OrdersRepository ordersRepository,
            boolean processAll) {

        this.view = view;
        this.filters = filters;
        this.ordersRepository = ordersRepository;
        this.processAll = processAll;

        view.setPresenter(this);
        setupFab();
    }
```

```
@Override
public void start() {
    setupOrders();
}

@Override
public void onResultNext() {
    ++current;
    processNext();
}

@Override
public void onResultClose() {
    current = -1;
    processNext();
}

@Override
public void onOrderClicked(int orderId) {
    view.navigateToOrderDetails(orderId, 1, 1);
}

@Override
public void onFabCLick() {
    processNext();
}

private void setupOrders() {
    if (orders != null) {
        return;
    }

    current = 0;

    ordersRepository.getList(
            filters,
            new RepositoryCallback<List<Order>>() {
                @Override
                public void onSuccess(List<Order> result) {
                    orders = result;
                    if (result.size() == 0) {
                        view.showEmptyList();
                    } else {
                        view.showOrdersList(orders);
                    }
                }

                @Override
                public void onError(List<String> errors) {
                    if (errors != null) {
                        view.showError(errors.get(0));
                    } else {
                        view.showCommunicationError();
                    }
                }
            });
}

private void processNext() {
    if (current >= orders.size() ||
```

```
                current < 0) {

            orders = null;
            setupOrders();
            return;
        }

        view.navigateToOrderDetails(
                orders.get(current).id(),
                current + 1,
                orders.size());
    }

    private void setupFab() {
        if (processAll) {
            view.showFab();
        } else {
            view.hideFab();
        }
    }
}
```

B.3.1 Testes

Listagem B.93 – BarcodeProcessorActivityTest

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;


import android.app.AlertDialog;
import android.content.DialogInterface;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;
import org.robolectric.Robolectric;
import org.robolectric.RobolectricTestRunner;
import org.robolectric.annotation.Config;
import org.robolectric.shadows.ShadowActivity;
import org.robolectric.shadows.ShadowAlertDialog;
import org.robolectric.shadows.ShadowLooper;

import java.util.HashMap;
import java.util.Map;

import br.com.aaascp.gerenciadordepedidos.BuildConfig;
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;

import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertTrue;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.containsString;
import static org.mockito.Mockito.verify;
import static org.robolectric.Shadows.shadowOf;

/**
```

```
 * Created by andre on 09/10/17.
 */
@RunWith(RobolectricTestRunner.class)
@Config(constants = BuildConfig.class)
public class BarcodeProcessorActivityTest {

    @Mock
    private BarcodeProcessorContract.Presenter presenter;

    private BarcodeProcessorActivity activity;

    @Before
    public void setup() {
        MockitoAnnotations.initMocks(this);

        activity = Robolectric.setupActivity(BarcodeProcessorActivity.class);
        activity.setPresenter(presenter);
    }

    @Test
    public void onItemProcessed() {
        String code = "1234";
        activity.onItemProcessed(code);

        verify(presenter).onItemProcessed(code);
    }

    @Test
    public void setupToolbar() {
        int orderId = 1000;
        activity.setupToolbar(orderId);

        assertThat(
                activity.title.getText().toString(),
                containsString(String.valueOf(orderId)));
    }

    @Test
    public void setItemsLeft() {
        int itemsLeft = 2;
        activity.setItemsLeft(itemsLeft);

        assertThat(
                activity.itemsLeft.getText().toString(),
                containsString(String.valueOf(itemsLeft)));
    }

    @Test
    public void setZeroItemsLeft() {
        activity.setZeroItemsLeft();

        assertEquals(
                activity.itemsLeft.getText().toString(),
                activity.getString(R.string.barcode_processor_items_left));
    }

    @Test
    public void showProcessError() {
        activity.showProcessError();
```

```
        ShadowLooper.runUiThreadTasksIncludingDelayedTasks();
        verify(presenter).onProcessingDone();
    }

    @Test
    public void showProcessSuccess() {
        activity.showProcessSuccess();

        ShadowLooper.runUiThreadTasksIncludingDelayedTasks();
        verify(presenter).onProcessingDone();
    }

    @Test
    public void showFinishDialog() {
        activity.showFinishDialog();

        AlertDialog dialog = ShadowAlertDialog.getLatestAlertDialog();
        ShadowAlertDialog shadowDialog = shadowOf(dialog);

        assertEquals(
                shadowDialog.getTitle().toString(),
                activity.getString(R.string.barcode_processor_finish_dialog_title));
        assertEquals(
                shadowDialog.getMessage().toString(),
                activity.getString(R.string.barcode_processor_finish_dialog_message));

        dialog.getButton(DialogInterface.BUTTON_POSITIVE).performClick();
        verify(presenter).onFinish();
    }

    @Test
    public void close() {
        Map<String, Integer> codes = new HashMap<>(1);
        codes.put("1234", 1);

        CodesToProcess codesToProcess = CodesToProcess.create(codes, 1000);

        activity.close(codesToProcess);

        ShadowActivity shadowActivity = shadowOf(activity);

        assertEquals(
                BarcodeProcessorActivity.RESULT_OK,
                shadowActivity.getResultCode());

        assertEquals(
                shadowActivity.getResultIntent()
                        .getParcelableExtra(
                                BarcodeProcessorActivity.EXTRA_RESULT),
                codesToProcess);

        assertTrue(shadowActivity.isFinishing());
    }
}
```

Listagem B.94 – BarcodeProcessorPresenterTest

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.camera;

import org.junit.Before;
```

```
import org.junit.Test;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;

import java.util.HashMap;
import java.util.Map;

import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;

import static org.mockito.Mockito.times;
import static org.mockito.Mockito.verify;


/**
 * Created by andre on 09/10/17.
 */
public class BarcodeProcessorPresenterTest {

    @Mock
    private BarcodeProcessorContract.View view;

    private BarcodeProcessorPresenter presenter;


    private static CodesToProcess getCodesToProcess(
            String code,
            int itemsLeft) {

        Map<String, Integer> codes = new HashMap<>();
        codes.put(code, itemsLeft);

        return CodesToProcess.create(codes, 1000);
    }

    @Before
    public void setup() {
        MockitoAnnotations.initMocks(this);
    }

    private void init(CodesToProcess codesToProcess) {
        presenter = new BarcodeProcessorPresenter(view, codesToProcess);
    }

    @Test
    public void onConstruction_setPresenter() {
        CodesToProcess codesToProcess = getCodesToProcess("1234", 1);
        init(codesToProcess);

        verify(view).setPresenter(presenter);
    }

    @Test
    public void onStart_setupToolbar() {
        CodesToProcess codesToProcess = getCodesToProcess("1234", 1);
        init(codesToProcess);

        presenter.start();

        verify(view).setupToolbar(codesToProcess.orderId());
    }
```

```
@Test
public void onStart_setItemsLeft() {
    CodesToProcess codesToProcess = getCodesToProcess("1234", 1);
    init(codesToProcess);

    presenter.start();

    verify(view).setItemsLeft(codesToProcess.itemsLeft());
}

@Test
public void onStart_setItemsLeft_finish() {
    CodesToProcess codesToProcess = getCodesToProcess("1234", 0);
    init(codesToProcess);


    presenter.start();

    verify(view).setZeroItemsLeft();
}

@Test
public void onFinish() {
    CodesToProcess codesToProcess = getCodesToProcess("1234", 1);
    init(codesToProcess);

    presenter.onFinish();

    verify(view).close(codesToProcess);
}

@Test
public void onItemProcessed_invalidCode() {
    String code = "1234";
    String invalidCode = "1235";

    CodesToProcess codesToProcess = getCodesToProcess(code, 1);
    init(codesToProcess);

    presenter.onItemProcessed(invalidCode);

    verify(view).showProcessError();
    verify(view).showCodeInvalidMessage(invalidCode);
}

@Test
public void onItemProcessed_alreadyProcessedCode() {
    String code = "1234";
    int codeQuantity = 2;

    CodesToProcess codesToProcess = getCodesToProcess(code, codeQuantity);
    init(codesToProcess);

    for (int i = 0; i <= codeQuantity; i++) {
        presenter.onItemProcessed(code);
        presenter.onProcessingDone();
    }

    verify(view).showProcessError();
```

```
        verify(view).showCodeAlreadyProcessedMessage(code);
    }

    @Test
    public void onItemProcessed_success() {
        String code = "1234";
        int codeQuantity = 2;

        CodesToProcess codesToProcess = getCodesToProcess(code, codeQuantity);
        init(codesToProcess);

        presenter.onItemProcessed(code);

        verify(view).showProcessSuccess();
        verify(view).setItemsLeft(codesToProcess.itemsLeft());
        verify(view).showSuccessMessage(code);
    }

    @Test
    public void onItemProcessed_success_finish() {
        String code = "1234";
        int codeQuantity = 2;

        CodesToProcess codesToProcess = getCodesToProcess(code, codeQuantity);
        init(codesToProcess);

        for (int i = 0; i < codeQuantity; i++) {
            presenter.onItemProcessed(code);
            presenter.onProcessingDone();
        }

        verify(view, times(codeQuantity)).showProcessSuccess();
        verify(view, times(codeQuantity)).showSuccessMessage(code);
        verify(view).setZeroItemsLeft();
        verify(view).showFinishDialog();
    }

    @Test
    public void onItemProcessed_notReady_notProcess() {
        String code = "1234";
        int codeQuantity = 2;

        CodesToProcess codesToProcess = getCodesToProcess(code, codeQuantity);
        init(codesToProcess);

        presenter.onItemProcessed(code);
        presenter.onItemProcessed(code);

        verify(view, times(1)).showProcessSuccess();
    }
}
```

Listagem B.95 – OrderDetailsActivityTest

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.details;

import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.Intent;
```

```
import android.support.v7.widget.RecyclerView;
import android.view.View;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;
import org.robolectric.Robolectric;
import org.robolectric.RobolectricTestRunner;
import org.robolectric.annotation.Config;
import org.robolectric.shadows.ShadowActivity;
import org.robolectric.shadows.ShadowAlertDialog;

import java.util.HashMap;
import java.util.Map;

import br.com.aaascp.gerenciadordepedidos.BuildConfig;
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderItem;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.camera.BarcodeProcessorActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.factories.OrdersFactory;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.item.OrderItemActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list.OrdersListActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.util.EmptyStateAdapter;

import static junit.framework.Assert.assertEquals;
import static junit.framework.Assert.assertTrue;
import static org.hamcrest.MatcherAssert.assertThat;
import static org.hamcrest.Matchers.containsString;
import static org.mockito.Mockito.verify;
import static org.robolectric.Shadows.shadowOf;

/**
 * Created by andre on 05/10/17.
 */

@RunWith(RobolectricTestRunner.class)
@Config(constants = BuildConfig.class)
public class OrderDetailsActivityTest {

    private static final Order ORDER =
            OrdersFactory.createOrder(1000, true);

    @Mock
    private OrderDetailsContract.Presenter presenter;

    private OrderDetailsActivity activity;

    @Before
    public void setup() {
        MockitoAnnotations.initMocks(this);
        activity = Robolectric.setupActivity(OrderDetailsActivity.class);

        activity.setPresenter(presenter);
    }

    @Test
```

```
public void onActivityResult() throws Exception {
    activity.navigateToProcessor(ORDER.codesToProcess());

    ShadowActivity.IntentForResult intentResult =
            shadowOf(activity).getNextStartedActivityForResult();

    Map<String, Integer> codes = new HashMap<>();
    for (String code : ORDER.codesToProcess().codes().keySet()) {
        codes.put(code, 0);
    }
    CodesToProcess newCodesToProcess =
            CodesToProcess.create(codes, ORDER.id());

    Intent result = new Intent();
    result.putExtra(
            BarcodeProcessorActivity.EXTRA_RESULT,
            newCodesToProcess);

    shadowOf(activity).receiveResult(
            intentResult.intent,
            Activity.RESULT_OK,
            result);

    assertEquals(
            intentResult.requestCode,
            OrderDetailsActivity.REQUEST_CODE_PROCESS);
    verify(presenter).onProcessorResult(newCodesToProcess);
}

@Test
public void onStart() throws Exception {
    activity.onStart();
    verify(presenter).start();
}

@Test
public void hideClearButton() throws Exception {
    activity.hideClearButton();

    assertEquals(
            activity.toolbar.getMenu().findItem(R.id.menu_order_details_clear).isVisible(),
            false);
}

@Test
public void hideSkipButton() throws Exception {
    activity.hideSkipButton();

    assertEquals(
            activity.toolbar.getMenu().findItem(R.id.menu_order_details_skip).isVisible(),
            false);
}

@Test
public void updateCodesProcessed() throws Exception {
    activity.showOrder(ORDER, ORDER.codesToProcess());

    Map<String, Integer> codes = new HashMap<>();
    for (String code : ORDER.codesToProcess().codes().keySet()) {
        codes.put(code, 0);
```

```
        }
        CodesToProcess newCodesToProcess = CodesToProcess.create(codes, ORDER.id());

        activity.updateCodesProcessed(newCodesToProcess);

        OrderDetailsAdapter.ViewHolder holder;
        int i = 0;
        for (OrderItem item : ORDER.items().values()) {
            holder = (OrderDetailsAdapter.ViewHolder) activity.recyclerView
                    .findViewHolderForAdapterPosition(i);

            int itemsLeft = item.quantity() - newCodesToProcess.codes().get(item.code());
            String itemsLeftText =
                    String.format(
                            activity.getString(R.string.order_details_count_text),
                            itemsLeft,
                            item.quantity());

            assertEquals(
                    holder.code.getText().toString(),
                    item.code());
            assertEquals(
                    holder.quantity.getText().toString(),
                    itemsLeftText);
            assertEquals(
                    holder.description.getText().toString(),
                    item.description());

            i++;
        }
    }

    @Test
    public void setupTitle_unique() throws Exception {
        activity.setupTitle(1000, 1, 1);

        assertThat(
                activity.toolbar.getTitle().toString(),
                containsString("#1000"));
    }

    @Test
    public void setupTitle_processAll() throws Exception {
        activity.setupTitle(1000, 1, 2);

        assertThat(
                activity.toolbar.getTitle().toString(),
                containsString("#1000 (1/2)"));
    }

    @Test
    public void setShipType() throws Exception {
        activity.setShipType(ORDER.shipmentInfo().shipType());

        assertEquals(
                activity.shipTypeView.getValue(),
                ORDER.shipmentInfo().shipType());
    }

    @Test
```

```
public void setItemsLeft() throws Exception {
    activity.setItemsLeft(2, 1);

    assertEquals(
            activity.itemsLeftView.getText().toString(),
            "1 / 2");
}

@Test
public void showOrder() throws Exception {
    activity.showOrder(ORDER, ORDER.codesToProcess());

    OrderDetailsAdapter.ViewHolder holder;
    int i = 0;
    for (OrderItem item : ORDER.items().values()) {
        holder = (OrderDetailsAdapter.ViewHolder) activity.recyclerView
                .findViewHolderForAdapterPosition(i);

        int itemsLeft = item.quantity() - ORDER.codesToProcess().codes().get(item.code());
        String itemsLeftText =
                String.format(
                        activity.getString(R.string.order_details_count_text),
                        itemsLeft,
                        item.quantity());

        assertEquals(
                holder.code.getText().toString(),
                item.code());
        assertEquals(
                holder.quantity.getText().toString(),
                itemsLeftText);
        assertEquals(
                holder.description.getText().toString(),
                item.description());

        i++;
    }
}

@Test
public void showError() throws Exception {
    String error = "Error";
    activity.showError(error);

    RecyclerView recyclerView =
            (RecyclerView) activity.findViewById(R.id.order_details_recycler);

    EmptyStateAdapter.ViewHolder holder =
            (EmptyStateAdapter.ViewHolder) recyclerView.findViewHolderForAdapterPosition(0);

    assertEquals(
            holder.getMessage().getText().toString(),
            error);
}


@Test
public void showCommunicationError() throws Exception {
    activity.showCommunicationError();
    RecyclerView recyclerView =
```

```
            (RecyclerView) activity.findViewById(R.id.order_details_recycler);

    EmptyStateAdapter.ViewHolder holder =
            (EmptyStateAdapter.ViewHolder) recyclerView.findViewHolderForAdapterPosition(0);

    String message = activity.getString(R.string.error_communication);
    assertEquals(
            holder.getMessage().getText().toString(),
            message);
}

@Test
public void hideLabels() throws Exception {
    activity.hideLabels();

    assertEquals(
            activity.finishRoot.getVisibility(),
            View.GONE);
    assertEquals(
            activity.itemsLeftRoot.getVisibility(),
            View.GONE);
    assertEquals(
            activity.alreadyProcessedRoot.getVisibility(),
            View.GONE);
}

@Test
public void showFinishLabel() throws Exception {
    activity.showFinishLabel();

    assertEquals(
            activity.finishRoot.getVisibility(),
            View.VISIBLE);
}

@Test
public void showItemsLeftLabel() throws Exception {
    activity.showItemsLeftLabel();

    assertEquals(
            activity.itemsLeftRoot.getVisibility(),
            View.VISIBLE);
}

@Test
public void showAlreadyProcessedLabel() throws Exception {
    activity.showAlreadyProcessedLabel();

    assertEquals(
            activity.alreadyProcessedRoot.getVisibility(),
            View.VISIBLE);
}

@Test
public void navigateToDetails() throws Exception {
    activity.showOrder(
            ORDER,

            ORDER.codesToProcess());
```

```
        OrderDetailsAdapter.ViewHolder holder =
                (OrderDetailsAdapter.ViewHolder) activity.recyclerView
                        .findViewHolderForAdapterPosition(0);

        holder.root.performClick();

        ShadowActivity shadowActivity = shadowOf(activity);
        Intent intent = shadowActivity.getNextStartedActivity();

        assertEquals(
                intent.getParcelableExtra(OrderItemActivity.EXTRA_ITEM),
                ORDER.items().get(holder.code.getText().toString()));
    }

    @Test
    public void finishSkip() throws Exception {
        activity.finishSkip();
        ShadowActivity shadowActivity = shadowOf(activity);

        assertEquals(
                OrdersListActivity.RESULT_CODE_SKIP,
                shadowActivity.getResultCode());

        assertTrue(shadowActivity.isFinishing());
    }

    @Test
    public void finishClose() throws Exception {
        activity.finishClose();
        ShadowActivity shadowActivity = shadowOf(activity);

        assertEquals(
                OrdersListActivity.RESULT_CODE_CLOSE,
                shadowActivity.getResultCode());

        assertTrue(shadowActivity.isFinishing());
    }

    @Test
    public void finishOk() throws Exception {
        activity.finishOk();
        ShadowActivity shadowActivity = shadowOf(activity);

        assertEquals(
                OrdersListActivity.RESULT_CODE_OK,
                shadowActivity.getResultCode());

        assertTrue(shadowActivity.isFinishing());
    }

    @Test
    public void finishOkUnique() throws Exception {
        activity.finishOkUnique();
        ShadowActivity shadowActivity = shadowOf(activity);

        assertEquals(
                OrdersListActivity.RESULT_CODE_OK_UNIQUE,
                shadowActivity.getResultCode());

        assertTrue(shadowActivity.isFinishing());
```

```
}

@Test
public void onBackPressed() throws Exception {
    activity.onBackPressed();

    verify(presenter).onBackPressed();
}

@Test
public void showBackDialog() throws Exception {
    activity.showBackDialog();

    AlertDialog dialog = ShadowAlertDialog.getLatestAlertDialog();
    ShadowAlertDialog shadowDialog = shadowOf(dialog);

    assertEquals(
            shadowDialog.getTitle().toString(),
            activity.getString(R.string.order_details_back_dialog_title));
    assertEquals(
            shadowDialog.getMessage().toString(),
            activity.getString(R.string.order_details_back_dialog_message));

    dialog.getButton(DialogInterface.BUTTON_POSITIVE).performClick();
    verify(presenter).onBackDialogOk();
}

@Test
public void showSkipDialog() throws Exception {
    activity.showSkipDialog();

    AlertDialog dialog = ShadowAlertDialog.getLatestAlertDialog();
    ShadowAlertDialog shadowDialog = shadowOf(dialog);

    assertEquals(
            shadowDialog.getTitle().toString(),
            activity.getString(R.string.order_details_skip_dialog_title));
    assertEquals(
            shadowDialog.getMessage().toString(),
            activity.getString(R.string.order_details_skip_dialog_message));

    dialog.getButton(DialogInterface.BUTTON_POSITIVE).performClick();
    verify(presenter).onSkipDialogOk();
}

@Test
public void showClearDialog() throws Exception {
    activity.showClearDialog();

    AlertDialog dialog = ShadowAlertDialog.getLatestAlertDialog();
    ShadowAlertDialog shadowDialog = shadowOf(dialog);

    assertEquals(
            shadowDialog.getTitle().toString(),
            activity.getString(R.string.order_details_clear_dialog_title));
    assertEquals(
            shadowDialog.getMessage().toString(),
            activity.getString(R.string.order_details_clear_dialog_message));

    dialog.getButton(DialogInterface.BUTTON_POSITIVE).performClick();
```

```
        verify(presenter).onClearDialogOk();
    }

    @Test
    public void showCloseDialog() throws Exception {
        activity.showCloseDialog();

        AlertDialog dialog = ShadowAlertDialog.getLatestAlertDialog();
        ShadowAlertDialog shadowDialog = shadowOf(dialog);

        assertEquals(
                shadowDialog.getTitle().toString(),
                activity.getString(R.string.order_details_close_dialog_title));
        assertEquals(
                shadowDialog.getMessage().toString(),
                activity.getString(R.string.order_details_close_dialog_message));

        dialog.getButton(DialogInterface.BUTTON_POSITIVE).performClick();
        verify(presenter).onCloseDialogOk();
    }

    @Test
    public void navigateToProcessor() throws Exception {
        activity.navigateToProcessor(ORDER.codesToProcess());

        ShadowActivity shadowActivity = shadowOf(activity);
        ShadowActivity.IntentForResult intentResult =
                shadowActivity.getNextStartedActivityForResult();

        assertEquals(
                intentResult.intent.getParcelableExtra(
                        BarcodeProcessorActivity.EXTRA_CODES_TO_PROCESS),
                ORDER.codesToProcess());
        assertEquals(
                intentResult.requestCode,
                OrderDetailsActivity.REQUEST_CODE_PROCESS);
    }

    @Test
    public void onFabClick() throws Exception {
        View fab = activity.findViewById(R.id.order_details_fab);
        fab.performClick();

        verify(presenter).onFabClicked();
    }

    @Test
    public void onFinishedClick() throws Exception {
        View onFinishedClick = activity.findViewById(R.id.order_details_finish);
        onFinishedClick.performClick();

        verify(presenter).onFinishClicked();
    }

    @Test
    public void onAlreadyProcessedClick() throws Exception {
        View alreadyProcessedButton = activity.findViewById(R.id.order_details_processed);
        alreadyProcessedButton.performClick();

        verify(presenter).onAlreadyProcessedClicked();
```

```
    }

    @Test
    public void onShipTypeClick() throws Exception {
        activity.shipTypeView.performClick();

        verify(presenter).onShipTypeClicked();
    }
}
```

Listagem B.96 – OrderDetailsPresenterTest

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.details;


import org.junit.Before;
import org.junit.Test;
import org.mockito.ArgumentCaptor;
import org.mockito.Captor;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;

import java.util.Collections;
import java.util.HashMap;
import java.util.Map;

import br.com.aaascp.gerenciadordepedidos.entity.CodesToProcess;
import br.com.aaascp.gerenciadordepedidos.entity.CustomerInfo;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.ShipmentInfo;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.factories.OrdersFactory;
import br.com.aaascp.gerenciadordepedidos.repository.OrdersRepository;
import br.com.aaascp.gerenciadordepedidos.repository.callback.RepositoryCallback;
import br.com.aaascp.gerenciadordepedidos.util.DateFormatterUtils;

import static org.mockito.Matchers.any;
import static org.mockito.Matchers.anyInt;
import static org.mockito.Mockito.never;
import static org.mockito.Mockito.times;
import static org.mockito.Mockito.verify;

/**
 * Created by andre on 05/10/17.
 */

public class OrderDetailsPresenterTest {

    private static final Order ORDER_PROCESSED =
            OrdersFactory.createOrder(
                    1000,
                    ShipmentInfo.builder()
                            .shipType("Sedex")
                            .address("Address")
                            .build(),
                    CustomerInfo.builder()
                            .id(1)
                            .name("Customer")
                            .build(),
                    2,
                    DateFormatterUtils.getDateHourInstance().now(),
```

```
                DateFormatterUtils.getDateHourInstance().now());


    private static final Order ORDER_NOT_PROCESSED =
            OrdersFactory.createOrder(
                    2000,
                    ShipmentInfo.builder()
                            .shipType("Sedex")
                            .address("Address")
                            .build(),
                    CustomerInfo.builder()
                            .id(1)
                            .name("Customer")
                            .build(),
                    2,
                    "",
                    DateFormatterUtils.getDateHourInstance().now());

    private static final Order ORDER_NOT_PROCESSED_FINISH =
            OrdersFactory.createOrder(
                    3000,
                    ShipmentInfo.builder()
                            .shipType("Sedex")
                            .address("Address").build(),
                    CustomerInfo.builder()
                            .id(1)
                            .name("Customer")
                            .build(),
                    0,
                    "",
                    DateFormatterUtils.getDateHourInstance().now());


    @Mock
    private OrderDetailsContract.View view;

    @Mock
    private OrdersRepository ordersRepository;

    @Captor
    private ArgumentCaptor<RepositoryCallback<Order>>
            repositoryCallbackArgumentCaptor;

    private OrderDetailsPresenter presenter;

    @Before
    public void setup() {
        MockitoAnnotations.initMocks(this);
    }

    private void init(Order order, int total, int current) {
        presenter = new OrderDetailsPresenter(
                view,
                ordersRepository,
                order.id(),
                total,
                current);

        verify(view).setPresenter(presenter);
        verify(view).setupMenu();
```

```
        presenter.start();

        verify(ordersRepository).getOrder(
                anyInt(),
                repositoryCallbackArgumentCaptor.capture());

        repositoryCallbackArgumentCaptor.getValue().onSuccess(order);
    }

    private void init() {
        init(ORDER_PROCESSED, 1, 1);
    }

    private void init(int total, int current) {
        init(ORDER_PROCESSED, total, current);
    }

    private void init(Order order) {
        init(order, 1, 1);
    }

    @Test
    public void initForUnique() throws Exception {
        int total = 1;
        int current = 1;

        init(ORDER_PROCESSED, total, current);

        verify(view).setupTitle(ORDER_PROCESSED.id(), current, total);
        verify(view).setupBackToolbar();
    }

    @Test
    public void initProcessAll() throws Exception {
        int total = 3;
        int current = 2;

        init(ORDER_PROCESSED, total, current);

        verify(view).setupTitle(ORDER_PROCESSED.id(), current, total);
        verify(view).setupCloseToolbar();
    }

    @Test
    public void start_orderProcessed() throws Exception {
        init(ORDER_PROCESSED, 1, 1);

        verify(view).setShipType(ORDER_PROCESSED.shipmentInfo().shipType());
        verify(view).showOrder(ORDER_PROCESSED, ORDER_PROCESSED.codesToProcess());
        verify(view).hideClearButton();
        verify(view).showAlreadyProcessedLabel();
    }

    @Test
    public void start_orderNotProcessed_finish() throws Exception {
        init(ORDER_NOT_PROCESSED_FINISH, 1, 1);

        verify(view).setShipType(ORDER_NOT_PROCESSED_FINISH.shipmentInfo().shipType());
        verify(view).showOrder(
```

```
                ORDER_NOT_PROCESSED_FINISH,
                ORDER_NOT_PROCESSED_FINISH.codesToProcess());
        verify(view).showFinishLabel();
    }

    @Test
    public void start_orderNotProcessed_itemsLeft() throws Exception {
        init(ORDER_NOT_PROCESSED, 1, 1);

        verify(view).setShipType(ORDER_NOT_PROCESSED.shipmentInfo().shipType());
        verify(view).showOrder(
                ORDER_NOT_PROCESSED,
                ORDER_NOT_PROCESSED.codesToProcess());
        verify(view).showItemsLeftLabel();
        verify(view).setItemsLeft(
                ORDER_NOT_PROCESSED.size(),
                ORDER_NOT_PROCESSED.codesToProcess().itemsLeft());
    }

    @Test
    public void start_error() throws Exception {
        init();

        String error = "Error";
        repositoryCallbackArgumentCaptor.getValue()
                .onError(Collections.singletonList(error));

        verify(view).showError(error);
    }

    @Test
    public void start_communicationError() throws Exception {
        init();

        repositoryCallbackArgumentCaptor.getValue().onError(null);

        verify(view).showCommunicationError();
    }

    @Test
    public void onMenuCreated_finish() throws Exception {
        init(2, 2);

        presenter.onMenuCreated();

        verify(view).hideSkipButton();
    }

    @Test
    public void onMenuCreated_itemsLeft() throws Exception {
        init(3, 2);

        presenter.onMenuCreated();

        verify(view, never()).hideSkipButton();
    }

    @Test
    public void onBackPressed_orderProcessed() throws Exception {
        init(ORDER_PROCESSED);
```

```
        presenter.onBackPressed();

        verify(view).finishBack();
        verify(view, never()).showBackDialog();
    }

    @Test
    public void onBackPressed_orderNotProcessed() throws Exception {
        init(ORDER_NOT_PROCESSED);

        presenter.onBackPressed();

        verify(view).showBackDialog();
        verify(view, never()).finishBack();
    }

    @Test
    public void onInfoClicked() throws Exception {
        init(ORDER_PROCESSED);

        presenter.onInfoClicked();

        verify(view).navigateToDetails(ORDER_PROCESSED);
    }

    @Test
    public void onProcessorResult_itemsLeft() throws Exception {
        init(ORDER_NOT_PROCESSED);

        Map<String, Integer> codes = new HashMap<>();
        for (String code : ORDER_NOT_PROCESSED.items().keySet()) {
            codes.put(code, 1);
        }
        CodesToProcess newCodesToProcess =
                CodesToProcess.create(codes, ORDER_NOT_PROCESSED.id());

        presenter.onProcessorResult(newCodesToProcess);

        verify(view).updateCodesProcessed(newCodesToProcess);
        verify(view, times(2)).showItemsLeftLabel();
    }

    @Test
    public void onProcessorResult_finish() throws Exception {
        init(ORDER_NOT_PROCESSED);

        Map<String, Integer> codes = new HashMap<>();
        for (String code : ORDER_NOT_PROCESSED.items().keySet()) {
            codes.put(code, 0);
        }
        CodesToProcess newCodesToProcess =
                CodesToProcess.create(codes, ORDER_NOT_PROCESSED.id());

        presenter.onProcessorResult(newCodesToProcess);

        verify(view).updateCodesProcessed(newCodesToProcess);
        verify(view).showFinishLabel();
    }
```

```
@Test
public void onProcessorResult_alreadyProcessed() throws Exception {
    init(ORDER_PROCESSED);

    Map<String, Integer> codes = new HashMap<>();
    for (String code : ORDER_NOT_PROCESSED.items().keySet()) {
        codes.put(code, 0);
    }
    CodesToProcess newCodesToProcess =
            CodesToProcess.create(codes, ORDER_NOT_PROCESSED.id());

    presenter.onProcessorResult(newCodesToProcess);

    verify(view).updateCodesProcessed(newCodesToProcess);
    verify(view, times(2)).showAlreadyProcessedLabel();
}

@Test
public void onClearClicked() throws Exception {
    init();

    presenter.onClearClicked();

    verify(view).showClearDialog();
}

@Test
public void onSkipClicked() throws Exception {
    init();

    presenter.onSkipClicked();

    verify(view).showSkipDialog();
}

@Test
public void onCloseClicked() throws Exception {
    init();

    presenter.onCloseClicked();

    verify(view).showCloseDialog();
}

@Test
public void onBackDialogOk() throws Exception {
    init();

    presenter.onBackDialogOk();

    verify(view).finishBack();
}

@Test
public void onSkipDialogOk() throws Exception {
    init();

    presenter.onSkipDialogOk();

    verify(view).finishSkip();
```

```
}

@Test
public void onClearDialogOk() throws Exception {
    init();

    presenter.onClearDialogOk();

    verify(ordersRepository, times(2)).getOrder(
            anyInt(),
            any(RepositoryCallback.class));
}

@Test
public void onCloseDialogOk() throws Exception {
    init();

    presenter.onCloseDialogOk();

    verify(view).finishClose();
}

@Test
public void onFabClicked_checkPermission() throws Exception {
    init(ORDER_NOT_PROCESSED);

    presenter.onFabClicked();

    verify(view).checkPermissionForCamera();
}

@Test
public void onPermissionEnabled_nothingProcessed() throws Exception {
    init(ORDER_NOT_PROCESSED);

    presenter.onCameraPermissionEnabled();

    verify(view).navigateToProcessor(ORDER_NOT_PROCESSED.codesToProcess());
}

@Test
public void onPermissionEnabled_codesProcessed() throws Exception {
    init(ORDER_NOT_PROCESSED);

    String code = ORDER_NOT_PROCESSED.items().keySet().iterator().next();
    Map<String, Integer> codes = new HashMap<>(2);
    codes.put(code, ORDER_NOT_PROCESSED.items().get(code).quantity() - 1);
    CodesToProcess newCodesToProcess =
            CodesToProcess.create(codes, ORDER_NOT_PROCESSED.id());

    presenter.onProcessorResult(newCodesToProcess);
    presenter.onCameraPermissionEnabled();

    verify(view).navigateToProcessor(newCodesToProcess);
}

@Test
public void onFinishClicked() throws Exception {
    init(ORDER_NOT_PROCESSED);
```

```
        presenter.onFinishClicked();

        verify(ordersRepository).save(
                ORDER_NOT_PROCESSED.withProcessedAt(
                        DateFormatterUtils.getDateHourInstance().now()));
    }

    @Test
    public void onFinishClicked_unique() throws Exception {
        init(ORDER_NOT_PROCESSED, 1, 1);

        presenter.onFinishClicked();

        verify(view).finishOkUnique();
        verify(view, never()).finishOk();
    }

    @Test
    public void onFinishClicked_processAll() throws Exception {
        init(ORDER_NOT_PROCESSED, 3, 1);

        presenter.onFinishClicked();

        verify(view).finishOk();
        verify(view, never()).finishOkUnique();
    }

    @Test
    public void onAlreadyProcessedClicked() throws Exception {
        init(ORDER_PROCESSED);

        presenter.onAlreadyProcessedClicked();

        verify(view).finishClose();
    }

    @Test
    public void onShipTypeClicked() throws Exception {
        init(ORDER_PROCESSED);

        presenter.onShipTypeClicked();

        verify(view).navigateToDetails(ORDER_PROCESSED);
    }

    @Test
    public void checkFinish_alreadyProcessed() throws Exception {
        init(ORDER_PROCESSED);

        verify(view).hideLabels();
        verify(view).hideClearButton();
        verify(view).showAlreadyProcessedLabel();
        verify(view, never()).showFinishLabel();
        verify(view, never()).showItemsLeftLabel();
        verify(view, never()).setItemsLeft(anyInt(), anyInt());
    }

    @Test
    public void checkFinish_notProcessed() throws Exception {
        init(ORDER_NOT_PROCESSED);
```

```
        verify(view).hideLabels();
        verify(view).showItemsLeftLabel();
        verify(view).setItemsLeft(
                ORDER_NOT_PROCESSED.size(),
                ORDER_NOT_PROCESSED.codesToProcess().itemsLeft());
        verify(view, never()).hideClearButton();
        verify(view, never()).showAlreadyProcessedLabel();
        verify(view, never()).showFinishLabel();
    }

    @Test
    public void checkFinish_notProcessed_finish() throws Exception {
        init(ORDER_NOT_PROCESSED_FINISH);

        verify(view).hideLabels();
        verify(view).showFinishLabel();
        verify(view, never()).setItemsLeft(anyInt(), anyInt());
        verify(view, never()).hideClearButton();
        verify(view, never()).showAlreadyProcessedLabel();
        verify(view, never()).showItemsLeftLabel();
    }
}
```

Listagem B.97 – OrdersListActivityTest

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list;

import android.content.Intent;
import android.support.v4.content.ContextCompat;
import android.support.v7.widget.RecyclerView;
import android.view.View;

import org.junit.Before;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;
import org.robolectric.Robolectric;
import org.robolectric.RobolectricTestRunner;
import org.robolectric.annotation.Config;
import org.robolectric.shadows.ShadowActivity;

import java.util.Arrays;
import java.util.Collections;

import br.com.aaascp.gerenciadordepedidos.BuildConfig;
import br.com.aaascp.gerenciadordepedidos.R;
import br.com.aaascp.gerenciadordepedidos.entity.CustomerInfo;
import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.ShipmentInfo;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.details.OrderDetailsActivity;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.factories.OrdersFactory;
import br.com.aaascp.gerenciadordepedidos.presentation.util.EmptyStateAdapter;
import br.com.aaascp.gerenciadordepedidos.util.DateFormatterUtils;

import static org.junit.Assert.assertEquals;
import static org.mockito.Mockito.verify;
import static org.robolectric.Shadows.shadowOf;
```

```
/**
 * Created by andre on 02/10/17.
 */

@RunWith(RobolectricTestRunner.class)
@Config(constants = BuildConfig.class)
public class OrdersListActivityTest {

    private static Order ORDER_PROCESSED =
            OrdersFactory.createOrder(
                    1000,
                    ShipmentInfo.builder()
                            .address("Endere o")
                            .shipType("Sedex")
                            .build(),
                    CustomerInfo.builder()
                            .id(1)
                            .name("Customer")
                            .build(),
                    5,
                    DateFormatterUtils.getDateHourInstance().now(),
                    DateFormatterUtils.getDateHourInstance().now());

    private static Order ORDER_NOT_PROCESSED =
            OrdersFactory.createOrder(
                    1001,
                    ShipmentInfo.builder()
                            .address("Endere o")
                            .shipType("Transportadora")
                            .build(),
                    CustomerInfo.builder()
                            .id(1)
                            .name("Customer")
                            .build(),
                    3,
                    null,
                    DateFormatterUtils.getDateHourInstance().now());

    private OrdersListActivity activity;

    @Mock
    OrdersListContract.Presenter presenter;

    @Before
    public void setup() throws Exception {
        MockitoAnnotations.initMocks(this);
        activity = Robolectric.setupActivity(OrdersListActivity.class);

        activity.setPresenter(presenter);
    }

    @Test
    public void onStart_startPresenter() throws Exception {
        activity.onStart();
        verify(presenter).start();
    }

    @Test
    public void onActivityResult_ok() throws Exception {
        activity.navigateToOrderDetails(1000, 1, 2);
```

```
        Intent intent = shadowOf(activity).getNextStartedActivity();

        shadowOf(activity).receiveResult(
                intent,
                OrdersListActivity.RESULT_CODE_OK,
                null);

        verify(presenter).onResultNext();
    }

    @Test
    public void orderDetailsSkipResult_skip() throws Exception {
        activity.navigateToOrderDetails(1000, 1, 2);

        Intent intent = shadowOf(activity).getNextStartedActivity();

        shadowOf(activity).receiveResult(
                intent,
                OrdersListActivity.RESULT_CODE_SKIP,
                null);

        verify(presenter).onResultNext();
    }

    @Test
    public void orderDetailsSkipResult_okUnique() throws Exception {
        activity.navigateToOrderDetails(1000, 1, 1);

        Intent intent = shadowOf(activity).getNextStartedActivity();

        shadowOf(activity).receiveResult(
                intent,
                OrdersListActivity.RESULT_CODE_OK_UNIQUE,
                null);

        verify(presenter).onResultClose();
    }

    @Test
    public void orderDetailsSkipResult_close() throws Exception {
        activity.navigateToOrderDetails(1000, 1, 2);

        Intent intent = shadowOf(activity).getNextStartedActivity();

        shadowOf(activity).receiveResult(
                intent,
                OrdersListActivity.RESULT_CODE_CLOSE,
                null);

        verify(presenter).onResultClose();
    }

    @Test
    public void hideFab() throws Exception {
        activity.hideFab();

        assertEquals(activity.fab.getVisibility(), View.GONE);
    }
```

```
@Test
public void showFab() throws Exception {
    activity.showFab();

    assertEquals(activity.fab.getVisibility(), View.VISIBLE);
}

@Test
public void showOrdersList() throws Exception {
    activity.showOrdersList(
            Arrays.asList(ORDER_PROCESSED, ORDER_NOT_PROCESSED));

    OrdersListAdapter.ViewHolder holder =
            (OrdersListAdapter.ViewHolder) activity.recyclerView
                    .findViewHolderForAdapterPosition(0);

    assertEquals(
            holder.id.getText().toString(),
            String.valueOf(ORDER_PROCESSED.id()));
    assertEquals(
            holder.shipType.getText().toString(),
            ORDER_PROCESSED.shipmentInfo().shipType());
    assertEquals(
            holder.itemsCount.getText().toString(),
            String.valueOf(ORDER_PROCESSED.size()));
    assertEquals(
            holder.processedAt.getText().toString(),
            ORDER_PROCESSED.processedAt());
    assertEquals(
            holder.processedAt.getCurrentTextColor(),
            ContextCompat.getColor(activity, R.color.green));
    assertEquals(
            holder.lastModifiedAt.getText().toString(),
            ORDER_PROCESSED.lastModifiedAt());
    assertEquals(
            holder.action.getText().toString(),
            String.valueOf(activity.getString(R.string.orders_list_action_details)));

    holder = (OrdersListAdapter.ViewHolder) activity.recyclerView
            .findViewHolderForAdapterPosition(1);

    assertEquals(
            holder.id.getText().toString(),
            String.valueOf(ORDER_NOT_PROCESSED.id()));
    assertEquals(
            holder.shipType.getText().toString(),
            ORDER_NOT_PROCESSED.shipmentInfo().shipType());
    assertEquals(
            holder.itemsCount.getText().toString(),
            String.valueOf(ORDER_NOT_PROCESSED.size()));
    assertEquals(
            holder.processedAt.getText().toString(),
            activity.getString(R.string.order_list_processed_at_empty));
    assertEquals(
            holder.processedAt.getCurrentTextColor(),
            ContextCompat.getColor(activity, R.color.red));
    assertEquals(
            holder.lastModifiedAt.getText().toString(),
            ORDER_NOT_PROCESSED.lastModifiedAt());
    assertEquals(
```

```
            holder.action.getText().toString(),
            String.valueOf(activity.getString(R.string.orders_list_action_process)));
    }

    @Test
    public void showEmptyList() throws Exception {
        activity.showEmptyList();

        EmptyStateAdapter.ViewHolder holder =
                (EmptyStateAdapter.ViewHolder) activity.recyclerView
                        .findViewHolderForAdapterPosition(0);

        String message = activity.getString(R.string.order_list_empty);
        assertEquals(holder.getMessage().getText().toString(), message);
    }

    @Test
    public void showError() throws Exception {
        String error = "Error";
        activity.showError(error);

        EmptyStateAdapter.ViewHolder holder =
                (EmptyStateAdapter.ViewHolder) activity.recyclerView
                        .findViewHolderForAdapterPosition(0);

        assertEquals(
                holder.getMessage().getText().toString(),
                error);
    }


    @Test
    public void showCommunicationError() throws Exception {
        activity.showCommunicationError();

        EmptyStateAdapter.ViewHolder holder =
                (EmptyStateAdapter.ViewHolder) activity.recyclerView
                        .findViewHolderForAdapterPosition(0);

        String message = activity.getString(R.string.error_communication);
        assertEquals(
                holder.getMessage().getText().toString(),
                message);
    }


    @Test
    public void navigateToOrderDetails_unique() throws Exception {
        activity.showOrdersList(
                Collections.singletonList(ORDER_PROCESSED));

        OrdersListAdapter.ViewHolder holder =
                (OrdersListAdapter.ViewHolder) activity.recyclerView
                        .findViewHolderForAdapterPosition(0);

        holder.root.performClick();

        ShadowActivity shadowActivity = shadowOf(activity);
        ShadowActivity.IntentForResult intentForResult =
                shadowActivity.getNextStartedActivityForResult();
```

```
        assertEquals(
                intentForResult.requestCode,
                OrdersListActivity.REQUEST_CODE_ORDER_PROCESS);
        assertEquals(
                intentForResult.intent.getIntExtra(
                        OrderDetailsActivity.EXTRA_ORDER_ID, Order.INVALID_ORDER_ID),
                ORDER_PROCESSED.id());
        assertEquals(
                intentForResult.intent.getIntExtra(OrderDetailsActivity.EXTRA_TOTAL, 0),
                2);
        assertEquals(
                intentForResult.intent.getIntExtra(OrderDetailsActivity.EXTRA_CURRENT, 0),
                1);
    }

    @Test
    public void navigateToOrderDetails_withParams() throws Exception {
        activity.navigateToOrderDetails(1000, 2, 3);

        ShadowActivity shadowActivity = shadowOf(activity);
        ShadowActivity.IntentForResult intentForResult =
                shadowActivity.getNextStartedActivityForResult();

        assertEquals(
                intentForResult.requestCode,
                OrdersListActivity.REQUEST_CODE_ORDER_PROCESS);
        assertEquals(
                intentForResult.intent.getIntExtra(
                        OrderDetailsActivity.EXTRA_ORDER_ID, Order.INVALID_ORDER_ID),
                ORDER_PROCESSED.id());
        assertEquals(
                intentForResult.intent.getIntExtra(OrderDetailsActivity.EXTRA_CURRENT, 0),
                2);
        assertEquals(
                intentForResult.intent.getIntExtra(OrderDetailsActivity.EXTRA_TOTAL, 0),
                3);
    }

    @Test
    public void onFabClick() throws Exception {
        activity.onFabClick();
        verify(presenter).onFabCLick();
    }
}
```

Listagem B.98 – OrdersListPresenterTest

```
package br.com.aaascp.gerenciadordepedidos.presentation.ui.order.list;

import com.google.common.collect.Lists;

import br.com.aaascp.gerenciadordepedidos.entity.Order;
import br.com.aaascp.gerenciadordepedidos.entity.OrderFilterList;
import br.com.aaascp.gerenciadordepedidos.presentation.ui.order.factories.OrdersFactory;
import br.com.aaascp.gerenciadordepedidos.repository.OrdersRepository;
import br.com.aaascp.gerenciadordepedidos.repository.callback.RepositoryCallback;

import org.junit.Before;
import org.junit.Test;
```

```
import org.mockito.ArgumentCaptor;
import org.mockito.Captor;
import org.mockito.Mock;
import org.mockito.MockitoAnnotations;

import java.util.ArrayList;
import java.util.List;

import static org.mockito.Matchers.eq;
import static org.mockito.Mockito.verify;

/**
 * Created by andre on 02/10/17.
 */
public class OrdersListPresenterTest {

    private static int ORDERS_SIZE = 2;
    private static List<Order> ORDERS =
            OrdersFactory.getOrders(ORDERS_SIZE, 0.5);

    private static List<Order> ORDERS_EMPTY = new ArrayList<>(0);
    private static List<String> ERRORS = Lists.newArrayList("Error");
    private static List<String> ERRORS_EMPTY = null;

    @Mock
    private OrdersListContract.View ordersListView;

    @Mock
    private OrdersRepository ordersRepository;

    @Mock
    private OrderFilterList filters;

    @Captor
    private ArgumentCaptor<RepositoryCallback<List<Order>>>
            repositoryCallbackArgumentCaptor;

    private OrdersListPresenter ordersListPresenter;

    private void setProcessAllPresenter() {
        ordersListPresenter = new OrdersListPresenter(
                ordersListView,
                filters,
                ordersRepository,
                true);
    }

    private void setNotProcessAllPresenter() {
        ordersListPresenter = new OrdersListPresenter(
                ordersListView,
                filters,
                ordersRepository,
                false);
    }

    private void start(List<Order> orders) {
        ordersListPresenter.start();

        verify(ordersRepository).getList(
                eq(filters),
```

```
                repositoryCallbackArgumentCaptor.capture());

        repositoryCallbackArgumentCaptor
                .getValue()
                .onSuccess(orders);
    }

    private void startError(List<String> errors) {
        ordersListPresenter.start();

        verify(ordersRepository).getList(
                eq(filters),
                repositoryCallbackArgumentCaptor.capture());

        repositoryCallbackArgumentCaptor
                .getValue()
                .onError(errors);
    }

    @Before
    public void setupOrdersListPresenter() {
        MockitoAnnotations.initMocks(this);

        setProcessAllPresenter();
    }

    @Test
    public void onConstruction_setPresenter() {
        verify(ordersListView).setPresenter(ordersListPresenter);
    }

    @Test
    public void onConstruction_setupFab_processAll() {
        verify(ordersListView).showFab();
    }

    @Test
    public void onConstruction_setupFab_notProcessAll() {
        setNotProcessAllPresenter();
        verify(ordersListView).hideFab();
    }

    @Test
    public void onStart_setupOrders_success() {
        start(ORDERS);

        verify(ordersListView).showOrdersList(ORDERS);
    }

    @Test
    public void onStart_setupOrders_empty() {
        start(ORDERS_EMPTY);

        verify(ordersListView).showEmptyList();
    }

    @Test
    public void onStart_setupOrders_error() {
        startError(ERRORS);
```

```
        verify(ordersListView).showError(ERRORS.get(0));
    }

    @Test
    public void onStart_setupOrders_communicationError() {
        startError(ERRORS_EMPTY);

        verify(ordersListView).showCommunicationError();
    }

    @Test
    public void onOrderClicked_navigateToOrderDetails_withOrderId() {
        start(ORDERS);

        int orderId = ORDERS.get(0).id();
        ordersListPresenter.onOrderClicked(orderId);

        verify(ordersListView).navigateToOrderDetails(orderId, 1, 1);
    }

    @Test
    public void onResultNext_finish_reloadOrders() {
        start(ORDERS);

        ordersListPresenter.onResultNext();
        ordersListPresenter.onResultNext();

        verify(ordersListView).showOrdersList(ORDERS);
    }

    @Test
    public void onResultNext_notFinish_navigateToOrderDetails() {
        start(ORDERS);

        ordersListPresenter.onResultNext();

        verify(ordersListView).navigateToOrderDetails(
                ORDERS.get(1).id(),
                2,
                ORDERS_SIZE);
    }

    @Test
    public void onResultClose() {
        start(ORDERS);

        ordersListPresenter.onResultClose();
        verify(ordersListView).showOrdersList(ORDERS);
    }
}
```