VVS Asssignment 2 Report

Henrique Catarino-56278

1.Bugs Corrigidos

VAT com Espaços em Branco Antes e/ou Depois(múltiplos Controllers, inclusive o getCustomerPageController)

Na implementação original, o número VAT era obtido diretamente da requisição HTTP, sem qualquer tratamento para remoção de espaços em branco. Isso poderia levar a problemas ao tentar buscar ou manipular dados, já que os espaços em branco adicionais fazem com que o VAT não seja reconhecido corretamente no banco de dados. Para resolver este problema, implementei uma correção que remove os espaços em branco antes e depois do número VAT utilizando um trim(). Esta alteração garante que o VAT seja sempre tratado de forma consistente, independentemente de como foi inserido pelo User.

```
try {
    String vat = request.getParameter("vat").trim();
    String address = request.getParameter("address");
    String door = request.getParameter("door");
    String postalCode = request.getParameter("postalCode");
    String locality = request.getParameter("locality");
```

Prevenção de Addresses Duplicadas (GetCustomerPageController)

Foi identificado que a aplicação permitia a inserção de moradas exatamente iguais para um cliente, o que não é desejável. Para solucionar este problema, adicionei uma verificação que compara a nova Address com as existentes. Caso a Address já exista com os mesmos detalhes (endereço, porta, código postal e localidade), a aplicação não permitirá a inserção, prevenindo assim a inserção de addresses duplicadas duplicatas na base de dados mantendo assim a sua integridade.

```
if(address != null) {
    AddressesDTO allAddresses = cs.getAllAddresses(vatNumber);
    List<AddressDTO> addressList = allAddresses.addrs;
    boolean repeated = false;

    if(addressList.size() > 0) {

        for(AddressDTO addr : addressList) {
            if((addr.address.trim()).equals(((address + ";" + door + ";" + postalCode + ";" + locality)).trim())){
                repeated = true;
            }
        }

        if(repeated) {
            ch.addMessage("It was not possible to fulfill the request: ");
        }else {
            cs.addAddressToCustomer(vatNumber, (address + ";" + door + ";" + postalCode + ";" + locality));
        }
```

Validação de Telefones Repetidos (addCustomerPageController e UpdateCustomerContactsPageController)

Implementei uma validação que impede tanto a criação quanto a atualização dos números de telefone de Customers que já estejam a ser utilizados por outros Customers. Isto garante que cada número de telefone seja único no sistema, evitando confusões e possíveis problemas de contato.

Restrições para Adicionar Deliverys a Sales Fechadas(AddSaleDeliveryPageController)

Outro problema identificado foi a possibilidade de adicionar deliverys a sales que já se encontram fechadas. Para resolver isto, adicionei uma verificação que impede a adição de deliverys se a Sale estiver marcada como fechada. Esta alteração garante que o estado de uma sale seja respeitado, mantendo a lógica de negócios consistente e antecipando erros operacionais.

```
if(addr != null && sale != null) {
    if (isInt(sdh, sale, "Invalid Sale Id") && isInt(sdh, addr, "Invalid Address Id")) {
        int addr_id = intValue(addr);
        int sale_id = intValue(sale);

        boolean closed = false;
        List<SaleDTO> customerSales = ss.getAllSales().sales;

        for(SaleDTO theSale : customerSales) {
            if(theSale.id == sale_id && theSale.statusId.equals("C") ) {
                closed = true;
            }

        }

        if(closed == true) {
            sdh.addMessage("Esta sale está fechada");
            request.getRequestDispatcher("SalesDeliveryInfo.jsp").forward(request, response);
        }else {
            int customerVat = ss.addSaleDelivery(sale_id, addr_id);
            SalesDeliveryDTO sdd = ss.getSalesDeliveryByVat(customerVat);
            sdh.fillWithSalesDelivery(sdd.sales_delivery);
            request.getRequestDispatcher("SalesDeliveryInfo.jsp").forward(request, response);
        }
```

Proibição de Nomes de Customers Vazios (addCustomerPageController)

Finalmente, foi observado que a aplicação permitia a criação de Customers com nomes vazios, o que é inadequado. Implementei uma verificação que impede a criação de um cliente caso o nome fornecido esteja vazio. Esta alteração assegura que todos os Customers tenham um nome válido e evita registos incompletos no sistema.

```
if(designation.isEmpty()) {
    repeated = true;
}
```

2.Problema Identificado: Vendas Não São Removidas ao Eliminar um Customer

Ao realizar o teste da base de dados de verificar se ao eliminar o cliente as sales associadas ao mesmo são removidas. Isto deve-se, pois, na implementação atual do sistema a base de dados não tem a integridade referencial configurada corretamente. Para garantir que as vendas sejam removidas automaticamente adicionar chaves estrangeiras com a opção 'ON DELETE CASCADE' nas tabelas 'SALE' e 'SALEDELIVERY'. De modo a assegurar que quando um cliente é removido da tabela 'CUSTOMER', todas as vendas associadas na tabela 'SALE' todas as entregas na tabela 'SALEDELIVERY' também sejam removidas automaticamente. Deve-se fazer um procedimento semelhante para remover a Address e a saleDeliverys associa a esse Customer.

```
FOREIGN KEY (CUSTOMER_VAT) REFERENCES CUSTOMER(VATNUMBER) ON DELETE CASCADE
```

```
FOREIGN KEY (CUSTOMER_VAT) REFERENCES CUSTOMER(VATNUMBER) ON DELETE CASCADE,
```

3.Notas para os testes

Somente mencionar que de modo aos testes html correrem múltiplas vezes pela ordem que quiser o VAT é criado aleatoriamente, assim como o número de telefone e a porta e código postal na address. Apesar de a chance ser mínima uma vez que o número de combinações possíveis é elevado pode ocorrer no teste de inserir 2 addresses inserir repetidas uma vez que são sempre inseridas num customer default. No teste da saleDelivery um customer novo é sempre criado, mas existe a chance de o número de telefone ser repetido, novamente uma chance mínima.

4.Mock

Para responder à questão no relatório, não é possível fazer Mock em alguns módulos da camada de negócio, nomeadamente nos serviços. Uma vez estes serviços na sua forma original, i.e da forma como nos são entregues. Se utilizarmos como exemplo o Customer Service o facto de instanciar diretamente suas dependências (e.g., CustomerFinder, CustomerRowDataGateway e AddressRowDataGateway) nos seus métodos implica que o esta serviço tem acoplamento forte (tight coupling) o que significa que durante o teste, não é possível substituir estas dependências por mocks o que impede realizar testes unitários isolados do CustomerService.

Para além disto o CustomerService está implementado como um enumerado singleton e o Mockito não suporta simulação de enumerados singleton diretamente devido à sua imutabilidade inerente e padrões de instanciação restritos.

De modo a fazer refactoring para tornar o CustomerService "mockable" (assim como os outros serviços que tem o mesmo formato) teríamos de transformar a classe em uma classe que pode ser instanciada em vez de um enumerado singleton e

introduzir injeções de dependências para fornecer instâncias ao CustomerService quando necessário o que injetar instâncias Mock para testes.

A classe “refactored encontra-se no projeto entregue com o nome CustomerServiceMock.

Para poder comprovar que a nova class era Mockable criei uma classe chamada UpdateCustomerContactsPageControllerMock, uma cópia da classe original, no entanto utiliza o novo CustomerService com as modificações. Por fim criei a classe teste denominada de MockTest que irá validar o comportamento da classe UpdateCustomerContactsPageControllerMock, mais especificamente o método processo que é responsável por lidar com os pedidos HTTP. Para o Mock teste realizei 2 testes no entanto irei explicar o primero, os egundo pose ser analisado na classe emncionada atrás.

1. testUpdateCustomerContactSuccess: Testa o cenário em que o contacto do Customer é atualizado com sucesso, verificando que o método updateCustomerPhone é chamado com sucesso, verifica se a solicitação está configurada com o atributo "mensagem" indicando sucesso e verifique se a solicitação foi encaminhada para success.jsp.