

Relatório Para o Segundo Projeto de VVS

BRUNO LIU FC56297



# HTMLUnit

Para esta parte do projeto foi utilizado o HTMLUnit para realizar os testes. Após a execução dos testes é garantido que o sistema volta ao estado que tinha antes de começar os testes.

Para isso foi necessário implementar 3 páginas extras:

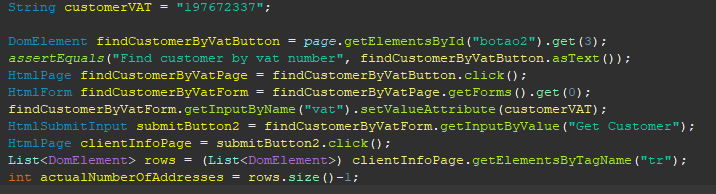
* DeleteLastSaleController.java -> Apaga a última sale inserida no sistema a um certo customer
* DeleteAllSaleDeliveryController.java -> Apaga todas as sale deliveries inserida no sistema a um certo customer
* DeleteAllAddressController.java -> Apaga todas os endereços inseridos no sistema a um certo customer

Foi necessário implementar estas 3 funcionalidades por o sistema não as possuía, estas funcionalidades são essenciais para garantir que o sistema volta ao seu estado base após a realização dos testes.

## a) Insert two new addresses for an existing customer, then the table of addresses of that client includes those addresses and its total row size increases by two

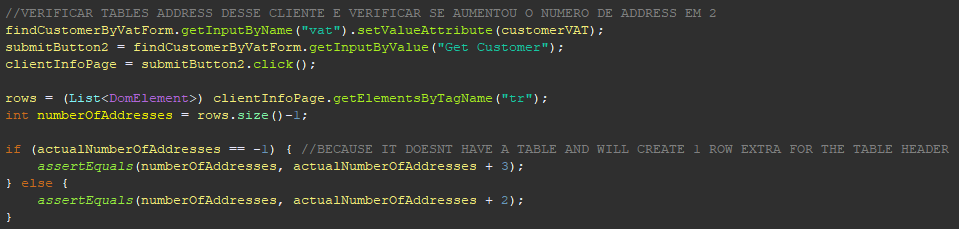
Foi escolhido um cliente já existente na base de dados, o “JOSE FARIA” cujo o seu VAT é 197672337.

Antes da introdução dos novos endereços para o cliente escolhido foi obtida a informação da invariante do sistema, que neste caso é o número de endereços existentes para esse cliente, atualmente, antes da realização dos testes. Este valor será utilizado para verificar se os endereços foram bem introduzidos e para verificar se os endereços foram bem retirados depois de voltar a remover esses endereços. O número de endereços foi obtido através do acesso à página “Find Customer by vat number”, onde são expostos os endereços do cliente cujo o vat lhe pertence.

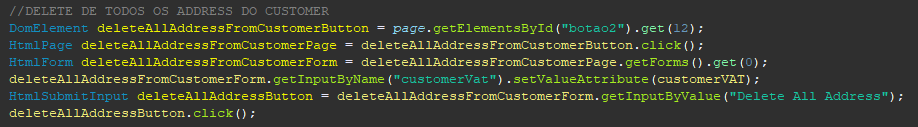


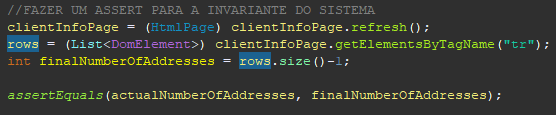
Após o número atual de endereços ser obtido foi feita a introdução de endereços através da página “Insert new Address to Customer”.

Após a introdução dos endereços ser feita, foi obtido o número total de endereços para esse certo cliente, mas agora após a realização do teste.

Depois de obter o valor é feita a comparação do número de endereços antes e depois da introdução de novos. É possível reparar que há um condicional, esse condicional existe, pois, para obter o número de endereços é contada as linhas da tabela, e se caso um cliente não tenha nenhum endereço associado essa tabela não existirá, isso significa que quando introduzido o primeiro endereço a quantidade de linha irá aumentar em 2, pois será introduzido o cabeçalho da tabela, que conta também como uma linha.

A seguir dos valores serem comparados é feita a remoção dos endereços através do acesso à nova página criada por mim.

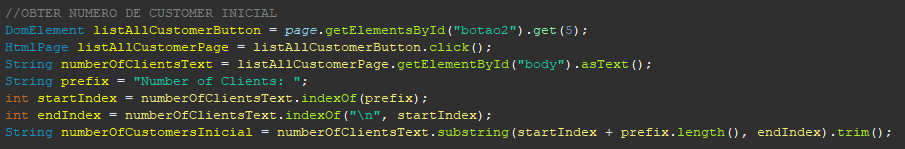
No final são comparadas as linhas de novo para verificar que o estado do programa antes da execução do teste se mantém.

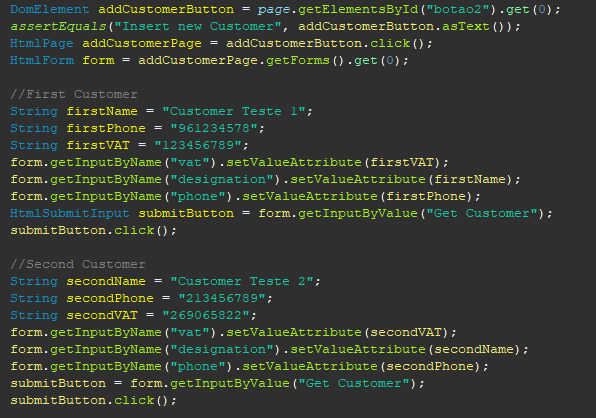


## b) Insert two new customers and check if all the information is properly

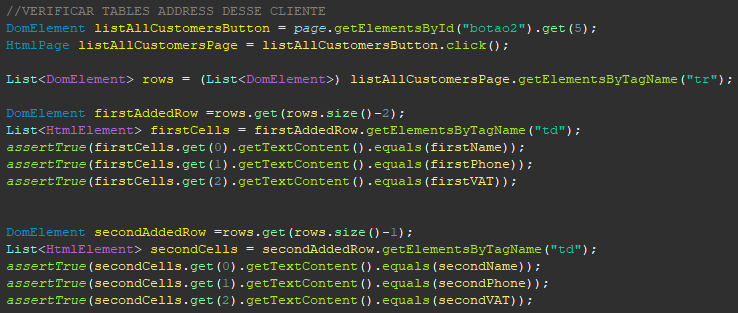
## listed in the List All Customers use case

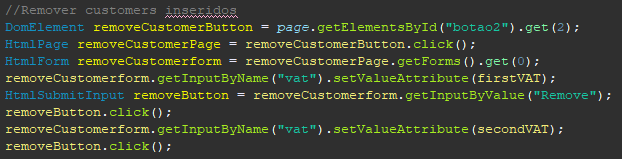
Antes da introdução de 2 novos clientes foi obtida a informação da invariante do sistema, que neste caso é o número total de clientes existentes atualmente, antes da realização dos testes. Este valor foi obtido através da página “List All Customers” e apenas será utilizado mais tarde para verificar se os clientes foram corretamente removidos.

Foi utilizada a página “Insert new Customer” e ao preencher os campos foram introduzidos 2 novos clientes.



Após a introdução dos 2 novos clientes a verificação da existência deles no sistema foi feita da seguinte maneira, foi acedida a página “List All Customers” e foi obtido as últimas 2 linhas da tabela, que iram corresponder aos 2 clientes inseridos, com essas 2 linhas foi feita a comparação de cada campo com a informação dos 2 clientes inseridos.

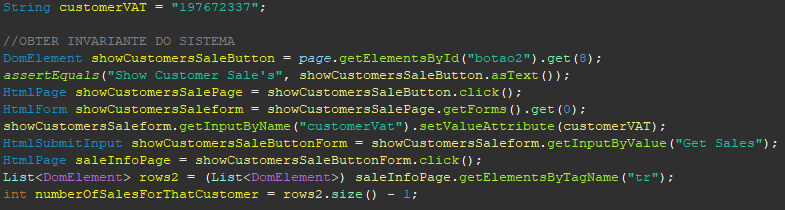
Foi utilizada a página “Remove Existing Customer” para remover os 2 clientes inseridos.

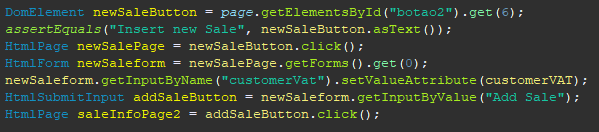
Após a remoção foi obtido o número total de clientes apos o teste e foi comparado com o valor obtido inicialmente e é verificado se são iguais.

## 

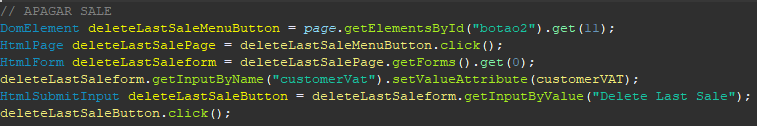
## c) A new sale will be listed as an open sale for the respective customer

Antes da introdução de 1 nova sale foi obtida a informação da invariante do sistema, que neste caso é o número total de sales existentes para um certo cliente, atualmente, antes da realização dos testes. Este valor foi obtido através da página “Show Customer Sale’s” e apenas será utilizado mais tarde para verificar se a sale foi corretamente removida.

Foi criada uma nova sale para um cliente já existente na base de dados por default, através da página “Insert New Sale”.

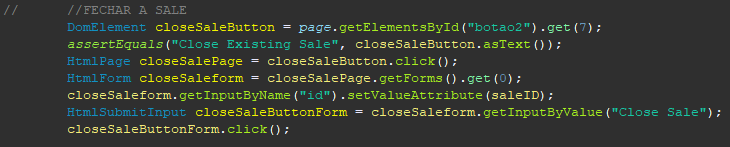
Após a sale ser criada sistema automaticamente reencaminha a página para uma pagina que contém todas as sales desse cliente, por essa pagina foi obtida a ultima linha, que corresponde à sale mais recente, e foi feita a comparação com os valores defaults que a sale fica quando é acabada de criar

Depois da comparação foi realizada a remoção da sale através da página “Delete Last Sale”, criada por mim.

No final foi obtido outra vez o número total de sales para comparar com o valor inicial e certificar que não mudou.

## d) After closing a sale, it will be listed as closed

Os passos para este teste são os mesmos até a inserção de uma sale nova, após essa inserção, é utilizada a página “Close Existing Sale” para fechar essa mesma sale.

Com a sale fechada é verificada através da tabela presente na página “Show Customer Sale”, a última linha da tabela, que corresponde a essa sale e é possível verificar o campo status e perceber que a sale está fechada, i.e, “C”.

Após isso os passos voltam a ser semelhantes aos passos presentes no teste anterior, é apagada a sale e é feita a verificação com o número de sales existentes.

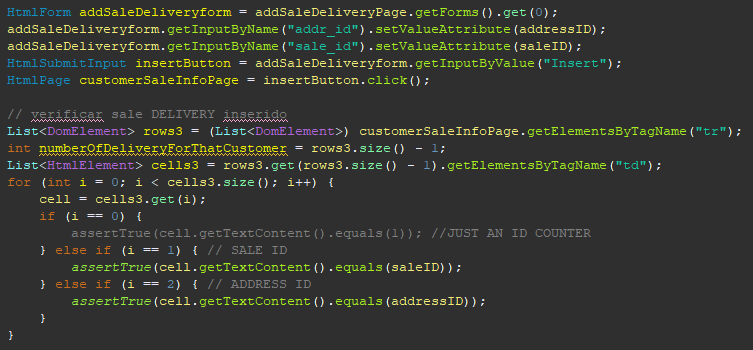
## e) Create a new customer, create a new sale for her, insert a delivery for that sale and then show the sale delivery. Check that all intermediate pages have the expected information.

Este teste é uma junção de todos os testes anteriores.

Um cliente é criado, é adicionado um endereço a esse cliente para depois ser utilizado no sale delivery. É criada uma sale para esse mesmo cliente e de em seguida é criada uma sale delivery com a id da sale criada e com o id do endereço que o cliente possui.

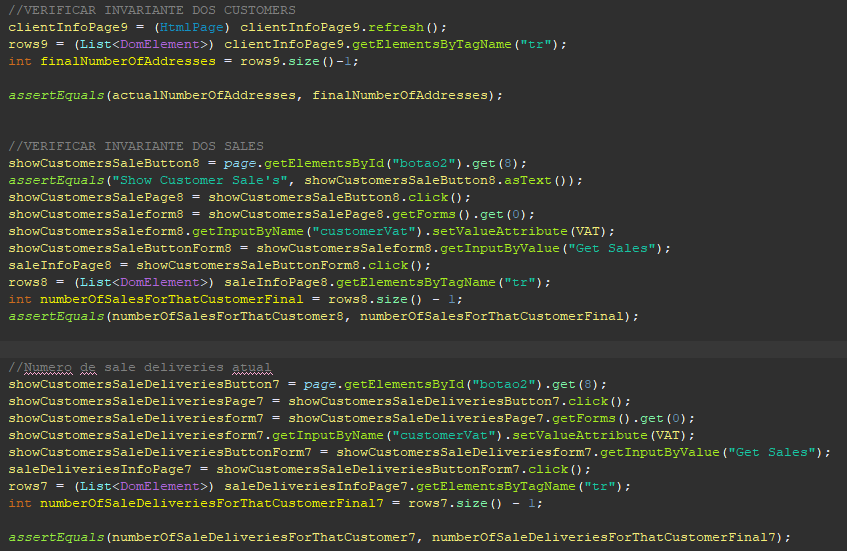
Entre estes passos todos é feita a verificação para perceber se os passos foram realizados com sucesso.

Depois de introduzir a sale delivery a mesma é verificada se foi bem introduzida:



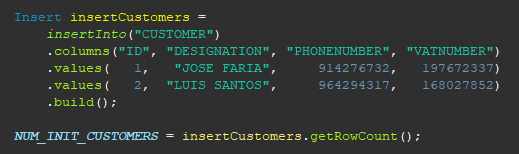
Após a verificação da sale delivery o teste está completo por isso só resta apagar tudo o que foi introduzido (customer, address, sale e sale delivery) e verificar a invariante do sistema.

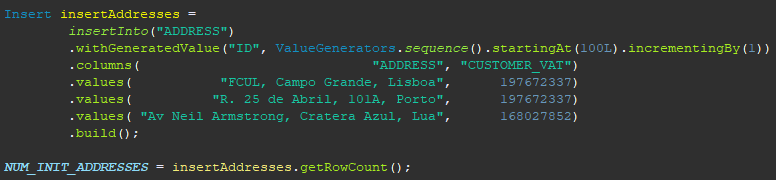




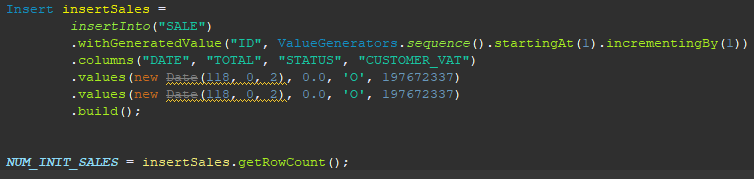
# DBSetup

Primeira mente precisamos de popular a base de dados com alguns valores para conseguir testar o código, os valores inseridos na base de dados foram:  
Para a tabela customer:

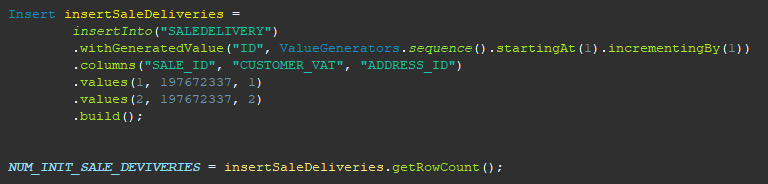


Para a tabela address:  


Para a tabela sale:

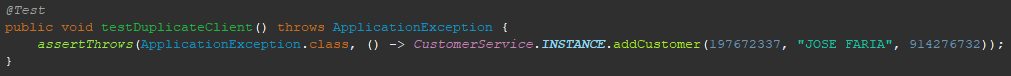


Para a tabela delivery:



## a) the SUT does not allow to add a new client with an existing VAT

Para este testei foi feito um assertThrows sobre a adição de um novo Customer.



## b) after the update of a customer contact, that information should be properly saved

É utilizado o CustomerService para alterar o número de telefone de um customer, após a alteração ser feita uma comparação é realizada para certificar que o uptade foi bem realizado

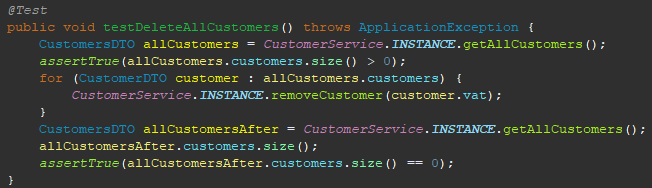
(PRINT)

## c) after deleting all costumers, the list of all customers should be empty

É feita a busca por todos os utilizadores, após obter a lista de utilizadores existentes é feita a remoção através do VAT 1 por 1. Após a remoção obtemos outra vez a lista de utilizadores e verificamos se essa lista está vazia

## d) after deleting a certain costumer, it’s possible to add it back without lifting exceptions

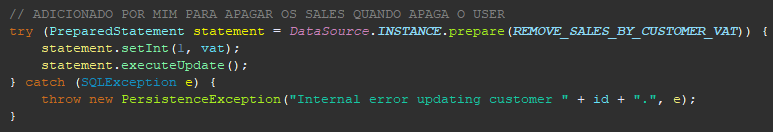
Para este teste é apagado um customer que já existe na base de dados e depois esse customer é voltado a ser inserido. No final faz se um getCustomerByVat através do CustomerService e são verificados os campos do customer para certificar que ele foi bem inserido de novo.



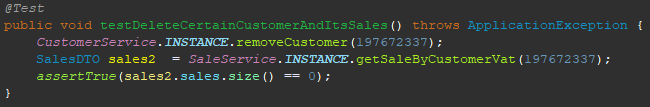
## e) after deleting a certain costumer, its sales should be removed from the database

Para a realização deste teste foi necessário implementar a deleção das sales de um certo customer quando esse customer é apagado.

Para a implementação da deleção das sales foi adicionado o seguinte código ao CustomerRowDataGateway.java:

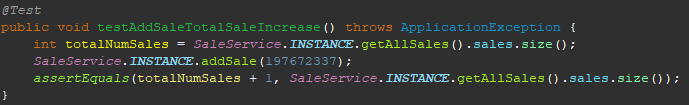
 

Após a deleção ser implementada, no teste é apagado um customer e é verificado se o número de sale desse customer estão a 0.



## f) adding a new sale increases the total number of all sales by one

Para este teste foi utilizado o SaleService para adicionar uma nova sale e foi comparado se o número total de sales aumentou.



## g) 2 extra tests concerning the expected behavior of sales

### 1. It is not possible to close a sale that doesn’t exist

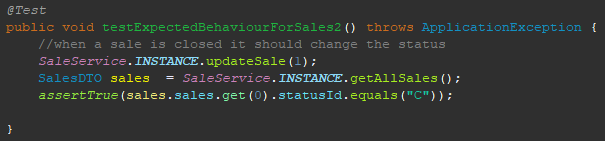
Foi utilizado um assertThrows para apanhar e exceção lancada pelo programa

### 

### 

### 2. When a sale is closed it should change the status

Mudamos o estado de uma sale já existente na base de dados e depois de mudarmos o estado da sale verificamos se o estado é igual a “C”, que significa Closed.



## h) 2 extra tests concerning the expected behavior of sales deliveries

### 1. After adding a sale delivery the number of sale deliveries should increase

Foi adicionada uma sale delivery a um certo customer e no final do teste é verificado se o número de sale deliveries aumentou em 1.

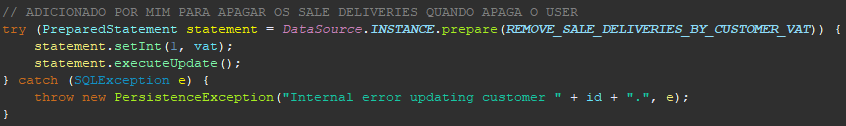
### 

### 2. After deleting a customer its sale deliveries should be deleted too

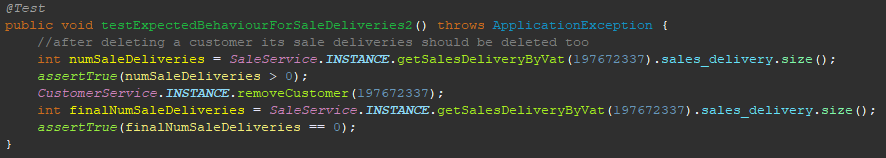
Para a realização deste teste foi necessário implementar a deleção das sales deliveries de um certo customer quando esse customer é apagado.

Para a implementação das sales deliveries foi adicionado o seguinte código ao CustomerRowDataGateway:





Após a deleção ser implementada, no teste é apagado um customer e é verificado se o número de sale deliveries desse customer estão a 0.

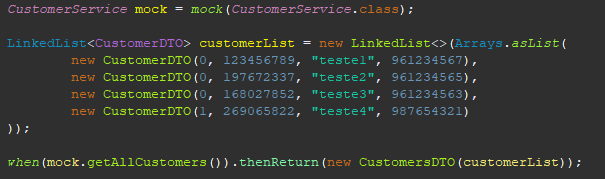


# Mockito

Neste projeto apenas temos 2 serviços e ambos são independentes uns dos outros. Mas existe uma dependência no que toca aos serviços e aos controllers, onde os controllers dependem do serviço para o seu funcionamento.

Logo numa primeira visão seria possível realizar um mock de um modulo da business layer para remover essa dependência entre serviços e controllers, mas da forma como o programa esta implementado as classes serviços, o CustomerService e o SaleService, são enumerados e em java um enumerado é implicitamente uma classe final, significando que não é possível realizar o mock de nenhum dos serviços existentes no projeto.

Para conseguir implementar o mocking no projeto seria necessário mudar o tipo da classe dos serviços para uma classe normal. Tendo os serviços como uma classe normal seria possível implementar o mock da seguinte forma:



Neste caso, o mock foi feito ao CustomerService e a único método onde foi feito o mock foi o getAllCustomers(), que é o método utilizado pelo GetAllCustomersPageController.java

# Bugs

Os bugs que foram encontrados ao longo dos testes realizado com o projeto foram:

* O espaço que fica se introduz um address a uma pessoa
* Se introduzir um address com ; a demonstração dele fica mal
* Depois de apagar um customer os sales ficam
* Depois de apagar um customer da para inserir sales para ele na mesma
* É possível começar uma delivery para um id sale que não existe
* É possível começar uma delivery para um id address que não existe
* Possível adicionar uma sale a um vat que não existe
* Introduzir um vat que não é valido a introduzir uma sale dá redirect para para o erro mas esse é introduzido na mesma uma sale para esse vat
* Quando adicionado uma sale ou sale delivery, se dermos f5 na pagina é possível introduzir uma nova sale ou sale delivery com as mesmas infos, pode ser uma coisa que pode ter sido deixada de propósito, seria necessário verificar os requisitos