# Teste de narrativas com HtmlUnit

Fazer testes diferentes para as mesmas narrativas? Um com o “Happy path” e outros com erros de utilização.

Há narrativas que requerem o uso de um cliente ja exsitente. No class start-up criar um cliente?

Os testes para cada uma das narrativas pedidas está no package htmlunit\_test

Foi criada uma class com métodos para as operações repetidas necessárias para as diferentes narrativas pedidas. Exemplos destas operações que se repetem são: criar um customer, remover um customer, criar address, remover um address, criar uma sale e remover uma sale.

É relevante mencionar que para cada narrativa no enunciado, foram feitos testes manuais de forma a se perceber o fluxo da troca de páginas, e o conteúdo de cada página. Também foram descobertos alguns bugs neste processo, preenchendo os formulários com valores nao adequados, sendo estes explorados em testes, devidamente documentados.

## Narrativa a)

Testes para a narrativa definida em 2.a) estão na classe InsertTwoAddressesToCustomerNarrativeTest.

## Narrativa b)

Testes para a narrativa definida em 2.b) estão na classe InsertTwoNewCustomersNarrativeTest

Testar meter o mesmo cliente

Dá para criar clientes com números de telefone impossíveis (valores negativos, poucos dígitos, demasiados digitos)

Os addresses são retornados pela bd com espaços em branco, no fim o que pode causar erros quando são feitas comparações da string de addresses vindos da bd

Sol: fazer trim das strings que são recebidas das bases de dados, na class AdressRowDataGateway.

## Narrativa c)

Testes para a narrativa definida em 2.c) estão na classe CreateSaleNarrativeTest.

## Narrativa d)

Testes para a narrativa definida em 2.d) estão na classe CloseSaleNarrativeTest.

## Narrativa e)

Testes para a narrativa definida em 2.e) estão na classe CreateDeliveryNarrativeTest.

Da para criar Delivery com address id que não existe?

Criar método de procura de address por id numa nova class AddressFinder

Em AddSaleDeliveryPageController chamar o método de procura do address\_id dado no request\_body, para garantir que ele existe. Caso não existir, este método vai lançar uma exceção, interrompendo o fluxo da função de inserção de delivery que seria o comportamento adqueado, em vez de deixar inserir deliveries com addresse\_id’s que não exsitem.

Construtor de AddressRowDataGateWay que recebe um ResultSet tinha nome da coluna customer\_vat mal

Antes tinha customerVat, mas o nome da coluna é customer\_Vat (é case insensivte)

Para além destes bugs, foram encontradas algumas funcionalidades que dependendo dos requisitos do cliente do software, podem ser determinados como bugs:

* É possível um customer criar uma delivery com um address id, de um Address que não foi criado por ele.
* É possível um customer criar uma delivery com um sale id, de uma Sale que não foi criada por ele.

## Reverter os inserts feitos nos testes para as narrativas

Modificou-se o SUT, para ter a capacidade de remover Adresses, Sales e Deliveries de Customers.

Para se poder remover Addresses acrescentou-se um link no index.html para se ser direcionado a um form de remoção de Address, parecido com o de inserção de um novo Address. A diferença é que este form faz um pedido para um outro PageController, criado por mim, o RemoveAddressPageController. Este controller recebe os dados necessários para a remoção de um Address de um Customer. Para seguir a logica de “separation of concerns” seguida pelos developers da app, foi criado o método removeAddressOfCustomer, na class CustomerService, método este que é então chamado no controller, abstraindo-o dos detalhes da remoção de um Address. Nesta função no CustomerService, é primeiro verificado que o customer identidicado com o vat dado tem o Address a remover, e no fim foi usada de novo a mesma lógica dos developers, sendo por isso usada AddressRowDataGateway, para tratar da lógica da camada dos dados.

Para se poder remover Sales, acrescentou-se um link no index.html para se ser direcionado para um form, que recebe o vat do customer, que criou a sale a apagar, e o id da sale a apagar. O pedido submetido por este form é tratado pelo RemoveSalePageController. Este controller lê os campos do pedido e usa o SalesService para se abstrair da lógica da remoção de uma sale, chamando o método removeSale com os valores do vat e id da sale lidos no pedido. No método removeSale, em SaleService, é verificado que o customer com o vat dado foi quem criou a sale, e é usada a class SaleRowDataGateway para se tratar da lógica de remoção de uma sale da base de dados.

Para se poder remover Deliveries acrescentou-se um link no index.html, para se ser direcionado para um form, que recebe o vat do customer, que criou a delivery, e o id da delivery a remover. Ao submeter um form é feito um pedido, que é tratado pelo controller RemoveDeliveryPageController. Neste controller são lidos os campos do pedido, e é usado o SaleService para abstrair o controller da remoção de uma delivery, com o método removeDelivery. Neste método, é primeiro verificado que o customer com o vat dado foi quem criou a Delivery , e é usado SaleDeliveryRowDataGateway para se tratar da lógica de remoção de Deliveries na base de dados.

Neste processo todo, como foram usados várias funcionalidades implementadas pelos developers, foi possível ter um melhor entendimento sobre os detalhes da implementação, o que tornou possível descobrir o seguinte bug:

Ponto 4. Mcokar um servisse tipo customer ou sale e teStar um controller?