1 Introduction

The goal of this second assignment is to apply the integration tools to test some *wepapp demo* functionalities.

The webapp demo is a layered application with:

- a persistence layer (using row data gateways to access the database);
- a business layer organized by services;
- a presentation layer following the MVC pattern with servlets as controllers, JSPs as views and java beans' helpers as business layer models. These beans are populated with information that come from the business layer using Data Transfer Objects (DTO's).

The business layer deals with four concepts: Customers, Addresses, Sales and Sales Deliveries. The presentation layer allows the user to execute the available use cases, like inserting new customers or close existing sales.

2 What to do

Your mission, should you choose to accept it, is to:

1. Participate on a shared project at backlog.com bugtracking system. This project should be shared by all elements of two to three groups, where you should include me as a member. Read details on the last slide of `vvs-slides-04-bugtracking`.
2. Use HtmlUnit to perform and test the following narratives in the webapp running on Wildfly:
   (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;
   (b) insert two new customers and check if all the information is properly listed in the *List All Customers* use case;
   (c) a new sale will be listed as an open sale for the respective customer;
   (d) after closing a sale, it will be listed as closed;
   (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.
3. Use DbSetup to populate the database with sample data. These data should be organized to test the following tasks:
   (a) the SUT does not allow to add a new client with an existing VAT;

(b) after the update of a costumer contact, that information should be properly saved;

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

(d) after deleting a certain costumer, it's possible to add it back without lifting exceptions;

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

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

Add two extra tests concerning the expected behaviour of sales and another two tests concerning sale deliveries.

4. Is it possible to mock some SUT business layer's modules (in this case, services) in order to remove dependencies and test these modules separately? If yes, pick two modules A and B with dependency $A \rightarrow B$, mock A and test B using Mockito. If not, explain why you cannot use Mockito as it is, and propose what kind of refactoring would the SUT's business layer need, in order to perform these mocking tests (show an example to explain your methodology).

If necessary, in order to implement what is asked in this assignment, you can adapt the SUT. In the final report, you should list all modifications and explain why was necessary to perform them.

If some failures occur, report them into your shared project at backlog.com. You should try to find and solve as many faults as possible into your individual project. The overall quality of the bug report made by all elements will also be part of the final evaluation of your own project.

3 Deliver

Upload a zip of your individual's work into the course's moodle webpage until 23:59 of May 28. The zip file should be named `vvs02_XXXX.zip` where `XXXXX` is your student's id number.

The zip must include:

- the maven project (with just the source code and `pom.xml`, no binaries or Eclipse metadata)
- a pdf report with all the necessary information regarding your work