ODSOFT 2022-2023 EDITION

PROJECT ASSIGNMENT – PART TWO

Hugo Silva (MBS)
Nuno Bettencourt (NMB)
Nuno Ferreira (NCF)
Nuno Melo (NLM)

Porto, November 17, 2022
Version 1.1

Index

1 CONTEXT

1.1 CONTENT MANAGEMENT SYSTEM - PRODUCT LINE

This project simulates the development of a proof-of-concept process and product for a hypothetical start-up company.

The project goal is to produce a proof-of-concept solution, state of the art in the field of Customer Relationship Management (CRM).

The start-up aims to innovate in this field by producing and implementing a software solution in short time frames, by applying a model-driven engineering approach on its software product line. The software product line must be supported by a continuous deployment approach.

This project concerns a Continuous Deployment Pipeline for ODSOFT Project Lab Assignment Part Two.

1.2 WEB APPLICATION CUSTOMER RELATIONSHIP MANAGEMENT (CRM)

The start-up will demonstrate the project to a group of investors in a near event (i.e., the final presentation of the project). In this event the start-up will present a web application developed using the innovative approaches presented earlier. This application should be based on the previously used project Vaadin-CRM.

In terms of features, this application must allow:

(i) Suppliers
   a. Suppliers are characterised by name, and address
   b. Suppliers must supply at least one category of products
   c. Administrators can add suppliers

(ii) Costumers
   a. Costumers are identified by first contact date, name, and address
   b. Customers can create Helpdesk Tickets

(iii) Helpdesk Tickets
   a. A Helpdesk Ticket is associated to a Costumer
   b. Helpdesk tickets contain an issue description and a reporting date

(iv) Product Categories
   a. Product categories have a description
   b. Administrators can create categories

2 CONTINUOUS DEPLOYMENT PIPELINE CONCERNS

The main goal of the assignment for ODSOFT is to continue the development of a continuous deployment pipeline in Jenkins with **declarative pipelines**.

This pipeline should support the development of the required features according to each concern.

The project must include the design and implementation for the provision of major concerns for the solution:

1. **Base Pipeline and Persistence.** This item must be cooperatively developed by all team members.
2. **Documentation and Database.** A team member is responsible for this concern, but it can be developed by more than one person. It is necessary to demonstrate clear evidence of individual work.
3. **Code Quality and Integration Tests.** A team member is responsible for this concern, but it can be developed by more than one person. It is necessary to demonstrate clear evidence of individual work.
4. **Functional and Smoke Testing.** A team member is responsible for this concern, but it can be developed by more than one person. It is necessary to demonstrate clear evidence of individual work.
5. **Non-Functional Testing.** A team member is responsible for this concern, but it can be developed by more than one person. It is necessary to demonstrate clear evidence of individual work.
6. **Continuous Deployment.** A team member is responsible for this concern, but it can be developed by more than one person. It is necessary to demonstrate clear evidence of individual work.

Issues should be created on Bitbucket for each of these concerns.

The Jenkinsfile should be able to be executed either on Windows or Linux operating systems.

Teams must always implement the first concern (as a team) and at least one individual concern for each team member, whereas **teams with more than one person must mandatorily implement concern #6.**

2.1 BASE PIPELINE AND PERSISTENCE

These are the requirements for the Base Pipeline and Persistence concern:

- The overall pipeline should be designed. This includes a high-level design for all the concerns;
- The design should include a description of the process, include the Git organization model (i.e., branching model) to be adopted and how it relates to or triggers the pipeline;
- The application should be completed with the specific features for the project;
- The application should have a persistence layer that uses a relational database. You may choose the technology to use for this non-functional requirement;
- Implement the pipeline for the specifics of this concern. The base pipeline implementation should include all the features addressed in Project Assignment - Part One, using parallel stages when suitable, even if they reside across different stages;
- Classify your final project according to the CI/CD Maturity Model. You should specify which level was achieved for each topic and justify the decision in the documentation.

2.2 DOCUMENTATION AND DATABASE

These are the base requirements for the Documentation and Database concern:

- The specific stages for this concern of the pipeline should be designed;
- The application should have a persistence layer that must use a relational database. You may choose the technology to use for this non-functional requirement. The application should use an SGBD running on a different system than the CRM web application using containerisation (e.g., Docker images). You may choose the technology to use for this non-functional requirement;
- Generate a Project Report PDF file from your project's Readme.md. You may choose the technology to use for this non-functional requirement;
- Generate and archive the project's Moodle zip submission file, containing the Project Report PDF, the Jenkinsfile and the generated artefacts for the build (e.g., Javadoc, reports, etc. – excluding all temporary files);

This is the advanced requirement for the Documentation and Database concern:

- The CRM should be adapted to foreseen third-parties version releases. In this case, explicitly document and adapt the current CRM solution for supporting Gradle's latest release (v. 7.5.2) and latest JDK 11 release (version may depend on the chosen vendor).
- You must use a migration tool to maintain database upgrades and downgrades, such as Flyway[1], Liquidbase[2] or other migration tool.

[1] https://flywaydb.org/
[2] https://www.liquibase.org/

2.3 CODE QUALITY AND INTEGRATION TESTS

These are the base requirements for the Code Quality and Integration Tests concern:

- The specific stages for this concern of the pipeline should be designed;
- The pipeline should include a "check" on the code quality of the project by using Checkstyle[3] or similar tool. Settings for code quality should be defined, including the thresholds for the build health. Analysis results must be published;
- The pipeline should include a "check" on the code quality of the project by using SpotBugs[4] or similar tool. Settings for code quality should be defined, including the thresholds for the build health. Analysis results must be published;
- Integration tests should cover the specific features of the project. Testing and Coverage Reports should be published. Settings for integration test coverage should be defined, including the thresholds for the build health. The build should fail if coverage degrades more than the delta threshold that should be configured;
- Analysis results must be published and explicitly discussed in the project report;

This is the advanced requirement for the Code Quality and Integration Tests concern:

- Test the CRM application at least against two different HTTP Servers versions (e.g., Tomcat v9.x, Tomcat v10.x). There should be a configuration file in the repository stating which versions of the server must be used and running the tests against more than two different versions should be supported. For this requirement Tomcat can be running locally.

[3] https://checkstyle.sourceforge.io/
[4] https://spotbugs.github.io/

2.4 FUNCTIONAL AND SMOKE TESTING

These are the base requirements for the Functional and Smoke Testing concern:

- The specific stages for this concern of the pipeline should be designed;
- Acceptance tests with Cucumber and Selenium should cover the specific features of the project. Cucumber Test and Coverage report should be published;
- These tests should be executed against an isolated staging environment (e.g, docker container) with the application – do not forget that the application may now be running on a different database!
- The image for this docker container should be published in docker hub;
- The smoke test should be executed against the execution of the previously built docker container with the application running – do not forget that the application may now be running on a different database!
- A report must be published on Jenkins.

This is the advanced requirement for the Functional and Smoke Testing concern:

- Run the CRM's acceptance tests using/simulating at least two different browsers. A configuration file in the repository should state which browsers and versions must be used to run the acceptance tests. More than two different browser/versions should be supported in the same build.

2.5 NON-FUNCTIONAL TESTING

These are the base requirements for the Non-Functional Testing concern:

- The specific stages for this concern of the pipeline should be designed;
- End 2 End (E2E) tests to evaluate non-functional requirements (i.e., system performance, throughput, and capacity) for two different usage scenarios (e.g., Suppliers and Helpdesk Tickets);
- A tool such as JMeter[5] (or any similar) should be used to measure those non-functional tests;
- These tests should be executed against an isolated staging environment (e.g, docker container) running the application – do not forget that the application may now be running on a different database!
- A report for must be published and archived on Jenkins.

This is the advanced requirement for the Non-Functional Testing concern:

- Run the CRM's non-functional requirements tests using at least two different HTTP servers (or major release versions of the same server). A configuration file in the repository should state which server and versions must be used to run the tests. More than two different server/versions should be supported in the same build.

[5] https://jmeter.apache.org/

2.6 CONTINUOUS DEPLOYMENT

These are the base requirements for the Continuous Deployment concern:

- The specific stages for this concern of the pipeline should be designed;
- This should be the last stage of the pipeline. The idea is to simulate a live deployment to production;
- This should simulate the production stage using a docker container;
- It should be able to deploy the binaries/resources and upgrade the database if necessary;
- It should be able to recover to the last "good" state if the upgrade fails;
- A tag should be created in the git repository related to the release and it also should be possible to identify this version from the deployed artefacts;

This is the advanced requirement for the Continuous Deployment concern:

- Perform a Blue/Green deployment of your application. For this task, the production environment should be running on at least three different machines e.g., a front-end tomcat proxy, the green production machine, and the blue production machine (do not forget that the application now has a database that should also be running on a different machine!).

3 TERMS AND CONDITIONS

3.1 DEADLINE

For this assignment, the last valid commit on the GIT repository should be dated before 22nd of December 2022 by 11.59 p.m.

3.2 PROJECT SUBMISSION

The Project Assignment submission includes the submission of the Project Report, the Jenkinsfile and the various generated technical artefacts, which must be present in the team's repository by the deadline and submitted to Moodle before the presentation date. The Project Report (PR) stands for 25% of the project score and should be written using a Markdown language. It should include:

- the details of the design;
- the technical details of the implementation;
- an analysis of possible alternatives and the justification for the options.

The artefacts in the repository must include the Jenkinsfile and the Project Report PDF. Each team **should** manually create a zip file containing the artefacts, Jenkinsfile and Project Report **only when** the zip file is not automatically created by the pipeline. The zip file must be submitted on Moodle before the presentation date by **all** team members.

3.3 GRADING

The project is assessed in the final presentation/demonstration session (D). The score of the project is calculated to the first decimal place and then converted to the scale 0-20,0.

A presentation/demonstration session (**D**) of the project will be scheduled. This stands for 75% of the project score. This session is mandatory and will take place on the last two weeks of the semester. If the assessment is performed using a video-conferencing session, video must be turned on for the whole session.

3.3.1 COMPONENTS

Each concern is graded using two dimensions: Requirements and Analysis. For Requirements (R) the rating scale is from 0 to 4, with the following semantics, that are adapted to suit the type of the assessment instrument:

- 0 - no submission;
- 1 - attempt, only some minor requirements;
- 2 - insufficiently achieves the requirements of the concern (50%);
- 3 - achieves the requirements of the concern with some minor faults;
- 4 - completely achieves all the requirements (basic and advanced) of the concern.

For Analysis (A) the rating scale is from 0 to 4, with the following semantics, that are adapted to suit the type of the assessment instrument:

- 0 - no analysis;

- 1 - attempt, analysis only contains very generic statements;
- 2 - insufficiently achieves the analysis, includes justification for some options (50%);
- 3 - partially achieves the analysis; includes justification for all the main options and some tentative of discussion of alternative(s);
- 4 - completely achieves, the analysis; includes justification for all the main options and a complete discussion of alternative(s).

The **score of the concern** is given by the following formula **R * 0,75 + A * 0,25**.

3.3.2 INDIVIDUAL / TEAM GRADING

Individual grading is performed according to the final presentation and peer feedback from students.

3.3.3 WORK EVIDENCE

Students should have evidence for: their development work (in commits to the repository) and pipeline implementation (based on the build history). This evidence is used for the grading.

3.4 ASSESSED OUTCOMES

This project contributes to the assessment of the following course outcomes:

1. Analyse how continuous delivery can address the difficulties of the software product life cycle management;
2. Organize the diverse components of a continuous delivery pipeline;
3. Plan tests, configuration management and versioning;
4. Manage a continuous delivery approach to software development;
5. Debate with colleagues about options in a continuous delivery project.

Therefore, it is expected that students attain these outcomes full/partially regarding the grading concerns.