Instituto Superior de Engenharia do Porto

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

# Lab Assignment MyRecipe

**ODSOFT and EDOM**

Master in Computer Science - 2016/2017

Porto, November 28, 2016

Alexandre Bragança (ATB),
Isabel Azevedo (IFP), Nuno Bettencourt (NMB)
António Silva Pereira (AMP)

`Version 0.3`

# Contents

# Chapter 1 Context

## 1.1 MyRecipe

MyRecipe is a startup company which goal is to provide to its users a service that allows them to share cooking recipes.

The company plans to provide its services via a web site and also via mobile applications (Android and iOS).

The company has selected OFBiz as a framework to support the development of its backend services and web site.

The company will present the project to a group of investors in a near event.

In this event the company will present a Console/Terminal application that will simulate a "regular" mobile application. This application will access the services of the OFbiz MyRecipe based server via REST web services.

For this event the company is only required to implement a continuous integration and deployment pipeline that should support the Console Application as well as the OFBiz server.

However the company is required to present to the investors a detailed plan on product releases and some technical details on how it will support the creation and maintenance (i.e., the life cycle) of the several applications with a continuous integration and delivery approach.

The Mobile application should have the following features:

- Recipes are private by default.
  - A recipe has ingredients with specific quantities
  - A recipe is of a type (e.g., bread, cake)
  - A recipe can have an image that illustrates its result.
  - A recipe has a sequence of steps. A step has a name and a textual description. A step can also have images that illustrate the operations involved in the step.
- Users can make their recipes public or share them with specific users (their friends)
- Users can vote on recipes.
  - When a user votes on a recipe he can also submit a small comment.
- User registration
- Login/Logout

From a technological point of view, MyRecipe Console Application is to be developed as a Java Console Application. The link between the client applications and backend services should be made using REST.

As stated before the company plan is to develop a mobile application in Objective C for iOS. The development of this application should start as soon as the investors provide the necessary funds. However, on the roadmap of the company, there is also a plan to start the development of a native Android within a one year time frame. The web server site should also start its development as soon as the iOS mobile application. Both iOS and site should be available in the next six months. The Java Console Application is to be maintained even after the event with the investors.

## 1.2 Use Cases

For the event with the investors only the following use cases are required in the Console Java Application and Backend (OFbiz).

### 1.2.1 "Register a Recipe"

The goal of this use case is for a logged user to register a new recipe.

- The system should display to the user the types of recipes available for the user to select one of them.
- The user should provide a name, one image (url) and notes related to the recipe.

- The user should register all the ingredients necessary to the recipe (and select the necessary unit of measure (UOM))
- The user should register all the steps of the recipe (description and one or more url of images)
- The user should select if the recipe is to be shared (for the moment the sharing is to all public) or not.

### 1.2.2 "View a Recipe"

The goal of this use case is for a logged user to view a recipe (his/her recipes or public recipes).

- The system should display to the user all his recipes as well as the the top 10 most voted public recipes.
- The user selects a recipe.
- The system displays the name, notes and image url of the selected recipe.
- If the recipe is public the user can choose to vote on the recipe (or change its previous vote) by selecting a integer number from 0 to 5 and a small comment.

**Note:** For the moment this use case does note include the display of the details of the recipe (its steps).

## 1.3 General Considerations

**Each group must only develop the components/requirements corresponding to course units in which they are enrolled.**

# Chapter 2 ODSOFT

## 2.1 Introduction

The main goal of the assignment for ODSOFT will be to implement a continuous integration pipeline with some requirements regarding automation of the application deployment.

The project must include the planning (roadmap) for the provision of 4 major components of the solution:

- Server application based on OFbiz
- Console "Mobile" Application
- IOS application
- Android Application

This plan should be described in a **technical report**. The technical report should two sections: one with the theoretical global plan and other related to the real plan that explains what will be implemented. The contents should be:

- The theoretical global plan (that **will not be implemented**):
  - The roadmap of the products
  - The design of the pipeline(s) with attention to dependencies between components/versions (it should include at least one adequate diagram)
  - The architecture of the solution (UML component diagram)
  - A deployment diagram illustrating the pipeline, in particular, the connection jenkins/bitbucket and other machines used in the envisioned solution (deployment, tests, etc)

  - A deployment diagram illustrating the solution, i.e., how all the products will connect (iOS, Android, web server, etc.)
  - The structure of the repository (it should include at least one adequate diagram)

- The real plan (that **will be implemented**):
  - The design of the pipeline (it should include at least an adequate diagram). You should not forget that we can only have one pipeline!
  - The architecture of the solution (UML component diagram)
  - The detailed design of the components of the pipeline (scripts, tool configuration, etc.)
  - Justification about the options and also a detailed analysis of the alternatives

You are only required to implement the Java Console Application and the Backend in OFBiz. In OFbiz you should implement the Entities, Services and REST web services required to support the two use cases that the Java Console Application must implement ("Register a Recipe" and "View a Recipe").

## 2.2 Pipeline

The pipeline must include all the topics covered during the ODSOFT classes. In particular, it must include at least:

1. The compilation of the applications/components
2. Unit tests and their code coverage
3. Integration tests and their code coverage
4. JMeter tests
5. Simulate a stage for automated acceptance tests (based on integration tests)
6. Simulate a stage for user acceptance tests in which the tests will be carried out in an environment similar to the end user environment. At this stage/phase you should also simulate the execution of UI tests performed manually by testers (one of the team elements should simulate a tester). For this step to succeed the tester should manually confirm the acceptance of the version being tested.
7. If all tests pass then it should be possible for a user to start the automatic process of deployment within the pipeline. This process should simulate the deployment of the applications to a new node. In this new node, after the necessary setup, a smoke test

should be performed to ensure that the deploy was successful. It should be issued a tag in the source control repository to mark the release.

In terms of final demonstration it is expected that the team prepares a new feature in a new branch of the repository. During the demonstration, this new feature should be integrated with the main branch of the repository and from that commit the team should demonstrate the previous described integration pipeline. The new feature must include tests of its functionality. The new feature can be the implementation of one of the required use cases (the other may already be in the main branch of the repository).

The team must prepare the demonstration so that it can take place during 15 to 20 minutes. If some steps of the pipeline take extensive time to execute then the team should include a variation of the pipeline that does not exceeds the time of the demonstration. This configuration may, for instance, use a subset of tests.

## 2.3 Advanced Issues

### 2.3.1 Data Migration

One of the situations that a continuous deployment pipeline must deal with is the necessity of data migrations when moving to a new release that requires changes in the structure or data of the existing database. You should describe a theoretical approach to deal with this issue. You can also demonstrate this issue for instance by illustrating how the pipeline can deal with the replacement of a column for another one that requires data migration.

### 2.3.2 Deployment Rollback

Other problematic issue is when the deployment of a version that passes all the tests fails. In this situation the pipeline should be able to rollback to the previous functioning version (you can assume the previous installed version was functioning). You should describe how to deal with this issue and simulate it during the demonstration.

## 2.4 Jenkins Server

There is a central Jenkins server that will be used as a central "verification" for the developed pipeline. The url for this server is `jenkins.dei.isep.ipp.pt`.

You should login to this server with the same credentials that you use for the portal of ISEP.

You will access an automatic generated job that follows the name of your team repository. This job is configured to execute the **Jenkinsfile** at the root of the master branch of your repository.

## 2.5 Terms and conditions

### 2.5.1 First Submission

In the first submission students must submit a first version of the technical report (**PR1**) that (at this moment) only needs to contain the theoretical global plan (see section 2.1) although it can be more complete (i.e., include also the **real** plan).

The technical report should be registered in the repository of the team in a suitable directory (please follow the instructions contained in the repository itself). The format of the technical report must be PDF.

The technical report has a weight of 20% in the project assessment. The technical report can be improved and completed in the final submission (**PR2**), but for this to be considered it is mandatory to have a submission at this stage. Otherwise, the report will not be considered in the final submission. The final submission of the report stands for 10% of the grade of the project.

#### Deadline

**Until the end of Dec 11.**

#### Feedback

The lab class Professor will provide feedback on this submission in one of the lab classes before the Christmas interruption. After this stage follows a design and implementation stage. This is due in the final submission date.

### 2.5.2 Final Submission

The final submission includes the submission of a technical report of the project (**FR**) and of the various technical artifacts, which must both be present in the repository of the team by that date.

The technical report may include the improvement of the plan (**PR2**) that was submitted previously. The technical report should also include the technical description of the design of the **real** solution (e.g., explaining the pipeline, the design of the server application, the build scripts, the design of the tests, etc.). Similarly to the weekly exercises, this report should also include the justification about the options and a detailed analysis of the alternatives.

The artifacts in the repository include the Jenkinsfile that should be tested using the Jenkins server of DEI.

After this, a presentation/demonstration session (**D**) of the project will be scheduled. This session is mandatory and should take place during the first two weeks of January.

#### Deadline

**Until 10:00am of January 2, 2017.**

### 2.5.3 Grading

The rating scales are from 0 to 4, with the following semantics (that are adapted to suit the type of the assessment instrument):

- 0- no submission
- 1- did not achieve the requirements of the deadline
- 2- achieved, partially, the requirements of the deadline
- 3- achieved, completely, the requirements of the deadline and the technical report includes justification about the options
- 4- achieved, completely, the requirements of the deadline and the technical report includes justification about the options and also a detailed analysis of the alternatives.

Levels 3 and 4 can only be achieved if the submission contains a technical report with the contents previously described.

There are 4 rubrics that use the previous rating scale: the first technical report with the plan (PR1); the improved plan report submitted in the final deadline (PR2); the final demonstration/defense session (D); and the final technical report (FR). Scores of the rubrics can be given to one decimal place. The score of the project is calculated to the first decimal place and then converted to the scale 0- 20,0. The formula is: PR1 * 0,2 + PR2 * 0,1 + D * 0,4 + FR * 0,3.

### 2.5.4 Assessed Outcomes

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

1. Analyze 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

### 2.5.5 Working Evidences

Students should have evidences of their work in commits in the repository. This evidences may be used in the grading.

# Chapter 3 EDOM

## 3.1 Introduction

The goal of the assignment for EDOM is to automate as much as possible the repetitive programming tasks required to implement the backend part of MyRecipe. To be more precise, the goal is to automate as much as possible the programming of OFbiz.

For that we will design and implement a textual DSL using EMF and EMFText. We will call this language "ofbiz domain model" (the file extension should be **ofbiz**).

**The main goal** is to use this language to define all the domain elements (entities, services, etc.) that are required to build the backend of MyRecipe. Basically we should be able to generate the following artifacts from scripts of our domain model language (i.e., instances of our DSL metamodel):

- The entity model of ofbiz (the files of type entitymodel.xml of OFbiz).
- The service model of ofbiz (the files of type services.xml of OFbiz).
  - It should also include the generation of java methods that implement the insert, update and delete operations for Entities.
  - This generation should be optional and executed on user request.
  - The generated code should be the "standard" for the type of operations and the user/developer should be able to customize it and keep the customization between generations.
- The java code for the REST services.

**As secondary goals** (but also very important) we will also want to:

- Generate class diagrams in papyrus from scripts of our domain model DSL in order to document the application. These class diagrams should depict essentially the Entities and their Relations and include the other aspects (services and REST) as comments/annotations in the diagram.

- Make our models as much as possible robust (therefore, a rich set of validations should be included in the metamodel)

The DSL should include the validation of the models (as much as possible). For example, it should not allow for invalid data types or invalid associations. It also should not allow specifying invalid java services (for example, invalid parameters in the method or specifying an invalid name for the java service).

Although the aim is that our DSL support as much as possible all that is possible to define in the entity and services models of OFbiz you are not required to make a complete "implementation". You may implement only the required options that allow the modelling of the MyRecipe backend. **Although, it should be interesting to demonstrate that the developed DSL is able to support existing entity model definitions of OFBiz (e.g., can it be used to reproduce an existing entitymodel.xml?).**

## 3.2 Technical Details Regarding OFbiz

**This section is focused on providing some initial guidance to students that are not enrolled in the ODSOFT course.**

You should start by reading the `readme` file in the `ofbiz` folder of your repository.
You may also find useful information in the following documents

- ODSOFT Lecture 02.1: Introduction to OFbiz

- ODSOFT Lecture 03.2: OFbiz REST Tutorial

- ODSOFT Lecture 04.1: Entities in OFbiz

- ODSOFT Lecture 04.2: Services in OFbiz

- "Entity Engine by Example" Appendix from "Apache OFBiz Cookbook"

- Service Engine Guide

- Lab Class 03: Exercise 2: Introduction to OFbiz (explains how to create a new component in OFbiz)

You can find the xsd with the specifics about entity and service definitions in:

- **ofbiz/framework/entity/dtd/entitymodel.xsd** (for entities)
- **ofbiz/framework/service/dtd/services.xsd** (for services)

You may find examples of the implementation of REST services in ofbiz in the folder **ofbiz/hot-deploy/restcomponent**. There you have the complete example of a REST service for the "PRODUCT" entity of OFBiz.

## 3.3 Terms and Conditions

### 3.3.1 First Submission

In the first submission students must submit a first version of the technical report (**PR1**) that (at this moment) only needs to contain a plan of the solution with some high level technical documentation such as:

- Activity diagram illustrating the envisioned solution and description of the major objects and tasks in the diagram (specially transformation tasks)
- Major concepts to be used in the metamodel of the new DSL (and justification based on the elements of the xsd files for entities and services as well as features of the REST code)

The technical report should be registered in the repository of the team in a suitable directory (please follow the instructions contained in the repository itself). The format of the technical report must be PDF.

The technical report has a weight of 20% in the project assessment. The technical report can be improved and completed in the final submission (**PR2**), but for this to be considered it is mandatory to have a submission at this stage. Otherwise, the report will not be considered in the final submission. The final submission of the report stands for 10% of the grade of the project.

#### Deadline

**Until the end of Dec 11.**

#### Feedback

The lab class Professor will provide feedback on this submission in one of the lab classes before the Christmas interruption. After this stage follows a design and implementation stage. This is due in the final submission date.

### 3.3.2 Final Submission

The final submission includes the submission of a technical report of the project (**FR**) and of the various technical artifacts, which must both be present in the repository of the team by that date.

The technical report may include the improvement of the plan (**PR2**) that was submitted previously. The technical report should also include the technical description of the design of the **real** solution (e.g., include diagrams for the metamodels, explaining the metamodels, explaining the transformations, etc.). Similarly to the weekly exercises, this report should also include the justification about the options and a detailed analysis of the alternatives. It is also necessary to include in the final technical report a section that should contextualize and justify why MyRecipe could/should be organized around a product-line approach.

The report should also explain how to execute and test the solution (e.g., what to to in eclipse).

After this, a presentation/demonstration session (**D**) of the project will be scheduled. This session is mandatory and should take place during the first two weeks of January.

#### Deadline

**Until 10:00am of January 2, 2017.**

### 3.3.3 Grading

The rating scales are from 0 to 4, with the following semantics (that are adapted to suit the type of the assessment instrument):

- 0- no submission
- 1- did not achieve the requirements of the deadline
- 2- achieved, partially, the requirements of the deadline
- 3- achieved, completely, the requirements of the deadline and the technical report includes justification about the options
- 4- achieved, completely, the requirements of the deadline and the technical report includes justification about the options and also a detailed analysis of the alternatives.

Levels 3 and 4 can only be achieved if the submission contains a technical report with the contents previously described.

There are 4 rubrics that use the previous rating scale: the first technical report with the plan (PR1); the improved plan report submitted in the final deadline (PR2); the final demonstration/defense session (D); and the final technical report (FR). Scores of the rubrics can be given to one decimal place. The score of the project is calculated to the first decimal place and then converted to the scale 0- 20,0. The formula is: PR1 * 0,2 + PR2 * 0,1 + D * 0,4 + FR * 0,3.

### 3.3.4 Assessed Outcomes

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

1. Analyze domain engineering as the top of an iceberg composed of a myriad of processes, techniques, tools and methodologies which all have reuse in software engineering as the ultimate goal
2. Analyze software product lines as one of the practices that promote large scale reuse
3. Design and implement domain-specific languages and model transformations in the context of model-driven engineering
4. Manage recent approaches in the context of domain engineering and reuse: model-driven software engineering and domain-specific languages
5. Debate with colleagues and teachers about options in projects related to domain engineering

### 3.3.5 Working Evidences

Students should have evidences of their work in commits in the repository. This evidences may be used in the grading.