Guided Exercise 3 Statement

Test Driven Development

Software Development

March 2023

Grado en Ingeniería Informática - Doble Grado en Informática y ADE

TABLE OF CONTENTS

OBJECTIVES

This guided exercise has three main goals:

- To practice the process of test-driven development.
- To learn the required tools of test-driven development.
- To practice the functional and structural testing techniques.

CASE STATEMENT

The specific goals for this exercise are to define, automatize and execute the required unit tests that will be used to verify whether the component for order management works correctly once its development is completed.

You have an initial version of the source code available in Aula Global. Notice that this component, at this moment, does not implement any functionality. The code is provided so that the execution of the test cases can be managed using the **PyBuilder** automation tool. However, it is expected that none of the test cases will work right since the design and implementation of the component's functionalities have not been done yet.

Students must develop the proposed functions based on the provided source code. They must apply the TDD techniques along with functional and structural testing techniques.

This component must be developed in Python 3.8, using the IDE PyCharm and configuring a virtual environment for the project. The code must be shared in a private repository on GitHub.

Functional Requirements

The functional requirements for defining the functional and structural test cases are as follows:

Function 1. Request the shipment of a product.

AM-FR-01: To manage the shipment of a product through the system, it must first have an order registered. The system will receive information about the order and will return a code that will be necessary to track it. This process is necessary to verify that the product codes received are correct.

Input:

AM-FR-01-I1: *product_id*. It is an EAN13 code that identifies the product that must be served.

AM-FR-01-I2: order_type. It can take the values "Regular" or "Premium". When the user is premium, the orders will be treated in a special way.

AM-FR-01-I3: address. Address to which the product is shipped. (between 20 and 100 characters with at least 2 strings separated by one white space)

AM-FR-01-I4: phone_number. Valid phone number. (9 digits)

AM-FR-01-I5: zip_code. A valid Spanish zip code.

Process:

AM-FR-01-P1: The component must verify the validity of the data received.

AM-FR-01-P2: If the data received is valid, the component will obtain a signature using the algorithm MD5. This MD5 value is obtained from the __str__ method of the

OrderRequest (class available in Aula Global). This signature will be the order identifier for the shipping process and will be called OrderID from now on.

AM-FR-01-P3: The system must store the data of each request with its corresponding output into a file for further processing.

Outputs:

- AM-FR-01-O1: A MD5 hexadecimal string value corresponding to the generated MD5 code.
- AM-FR-01-O2: A file in which the data of the order being registered in the system has been included.
- AM-FR-01-O3: An exception in the following cases:
    - The EAN13 code received is not valid or has not a valid format.
    - The type of order is not valid.
    - The address has an invalid format.
    - The phone number format is not valid.
    - The zip code is not valid.
    - Other internal errors.

Interface

The definition of the interface component that will provide this functionality is as follows:

```
register_order (product_id,
                order_type,
                address,
                phone_number,
                zip_code):
// Returns a string representing AM-FR-01-O1
// On errors, returns a VaccineManagementException according to
AM-FR-01-O2
```

Function 2. Sending a product.

AM-FR-02: To ship the product, the system must double-check the order identifier generated by the function above. If the order identifier is correct, the system will generate a tracking code that will be used by the transport company, with a date on which the product is expected to be delivered.

Inputs:

AM-FR-02-I1: The path of the file with the data of the product to be sent. The JSON input file must comply with the following format:

```
{

    "OrderID":"<String having 32 hexadecimal characters>",

    "ContactEmail": "<Valid email>"

}
```

Process

AM-FR-02-P1: the component must verify that the order request was stored in the request store and that the OrderID matches the data (that is, that the data in the file has not been manipulated).

AM-FR-02-P2: If the data received is valid, the component must generate an instance of the class OrderShipping. The attributes of that class are:

- "alg" (String). It identifies the algorithm used for signing the shipping code. By now, its value must be "SHA-256", but in future versions, the component will allow other values.
- "type" (String). By now, the component only allows the value "UC3M" for this field.
- "order_id" (String). This value is the value of the PatientSystemID in AM-FR-01.
- “issued_at”. Contains the shipping issue date (UTC time in timestamp format). This will be the current date.
- “delivery_day”. Product delivery date (UTC time in timestamp format). If the user who requested the product is "Premium" this date will be the following day, while

in the case of "Regular" this date will be 7 days later.

- “tracking_code” (String). This is the code that can be used to track the shipment during its transport. This code is calculated by encoding the above fields using “SHA-256”. The text to be encoded has the format:

```
{alg:<value>,typ:<value>,order_id:<value>,issued_at:<value>,
                    delievery_day:<value>}
```

AM-FR-02-P2: The component must store all correctly processed shipments into a file.

Outputs

AM-FR-02-O1: an SHA-256 hexadecimal string (the tracking number), if the order data received is correct (found in stored orders and data is ok).

AM-FR-02-O2: a file with all the correctly processed shipments.

AM-FR-02-O3: an AccessMangementException in the following cases:

- The input file is not found.
- The input file has no JSON format.
- The JSON has not the expected structure.
- The data of the JSON has no valid values.
- The order has not been found in the orders archive.
- Internal processing error when obtaining the signature.

Interface

The definition of the interface component that will provide this functionality is as follows:

```
send_product (input_file):

// The input file is a string with the file path described in AM-FR-02-I1

// Returns a String in hexadecimal which represents the tracking number (key that will be needed for AM-FR-02-O1 an AM-FR-02-02)

// In case of error, it returns an OrderManagementException described in AM-FR-02-O3
```

Function 3. Product Delivery

AM-FR-03: When the order reaches its destination, the system will record that it has been delivered to the customer. The system will verify that the tracking code is correct and that the day is the scheduled one (we will assume that it can only be delivered on the scheduled date). Finally, it will record the delivery into a file.

Inputs:

AM-FR-03-I1: A SHA256 hexadecimal string representing a tracking code generated by function 2.

Process:

AM-FR-03-P1: The component must verify the correctness of the tracking code received. Based on the experience of the previous methods, you must determine whether the signature received is valid or not.

AM-FR-03-P2: If the tracking code was valid the component should register into a file the timestamp (UTC time) of the delivery and the tracking code value.

Outputs

AM-FR-03-O1: The component will return true if the tracking code and the delivery date are valid.

AM-FR-03-O2: A file with the timestamp and the tracking code of all the orders successfully delivered.

AM-FR-03-O3: an OrderMangementException in the following cases:

- The input string does not contain a tracking code that can be processed.
- The tracking code to be verified is not registered.
- The tracking code is not valid (different values for each situation)
- Internal processing error when processing the tracking code (i.e. files not found, etc.)

Interface

The definition of the interface component that will provide this functionality is as follows:

```
deliver_product (tracking_code):

// the date_signature is a string with the value described in AM-FR-03-I1

// Returns a boolean value defined in AM-FR-03-O1 and a file defined in  AM-FR-03-O2

// On errors, returns a VaccineManagementException representing AM-RF -03-O3
```

TASKS TO DO

1. Create a new repository in GitHub for this guided exercise. Name this repository according to the rules described in guided exercise 2 (GXX.2023.TYY.EG3) and invite your lab teacher.
2. Clone de project and include the code and the folder structure available in Aula Global.
3. Following the TDD process, define the test cases and implement the first function. Apply the Equivalence Classes Analysis and the Boundary Values (when applicable) in this first method. Depending on the input and output parameters nature, several rules must be considered. Those rules are related to
    - Input/output values range
    - Amount of input values
    - Semantics of input values
4. Following the TDD process, define the test cases and implement the second function. In this case, apply the Syntax Analysis technique. If necessary, complement the tests by applying Equivalence Classes & Boundary Values.
5. Following the TDD process, define the test cases and implement the third function. Apply in this third case Structural testing techniques.
6. **The project must follow the PEP8 coding style**. You must check the code **using pylint with the default configuration**. All errors and warnings detected by pylint must be solved.
7. Both team members must interact with the GitHub repository, following the principles of pair programming and collective code ownership.
8. Create a PDF document and include the definition of the test cases for each function. For the second function, include also the grammar and the derivation tree. For the third function, include the Control Flow Graph, the definition of the

basic paths and the extra cases required for testing the loops. The look of the document will be considered in the mark (cover page, headings, the document's format, etc.)

Include also an excel file with the definition of the test cases, according to the template provided in Aula Global.

Both documents must be saved in a folder called "docs".

9. Generate a report (log) with the results of the execution of the tests for each function. Place these reports in the folder "docs\test_reports".

RULES AND PROCEDURES

This exercise will be solved in pairs. Once the exercise is finished, you should push in the corresponding branch of your project of the GitHub repository:

- The source code generated for each functionality during the TDD process.
- The code of the tests defined to automate the validation of selected test cases.
- The reports with the results of the execution of all the tests. These reports must be saved in a folder called "docs\test_reports".
- A PDF document containing:
    - The equivalence classes and boundary values analysis for functionality #1.
    - The grammar, the derivation tree and the selection of test cases to develop for functionality #2 using the syntax analysis technique.
    - The control flow graph, the basic paths and the additional cases for testing the loops for functionality #3 using the structural analysis technique.
    - This document must be saved in a directory called "docs".
- An excel file with the definition of the test cases according to the template

available in Aula Global. This document must be saved in a folder called "docs".

The tests defined in the Excel file for the first function will be sent by March 17th to another group (in aula global we will publish the list of groups to send the tests to). The students will make a cross corrections to their classmates.

All the tests and code will be delivered on March 31 along with a brief document commenting on the evaluation of the tests of corresponding classmates.

In accordance with the standards of continuous assessment established in this subject, if a team does not publish the solution of the exercise before the deadline, the exercise will be evaluated with a score of 0 points.