Diseño y Pruebas II

ANALYSIS REPORT

Grupo: C3.02.07
Repositorio: https://github.com/DP2-C1-02-07/Acme-L3
Autor: Julio Navarro Rodríguez - **julnavrod@alum.us.es**
Fecha: 23/10/2023

1. Índice

1. Índice 2
2. Tabla de versiones 3
3. Resumen del documento 3
4. Introducción 3
5. Contenidos 4
    5.1. Requisitos de información 4
    5.2. Requisitos funcionales 4
    5.3. Requisitos no funcionales 4
    5.4. Requisitos de testing 8
    5.5. Requisitos de gestión 8
6. Conclusiones 8
7. Bibliografía 8

2. Tabla de versiones

Versión	Fecha	Descripción
1.0.0	02/09/2023	Documento creado y requisitos incluidos

3. Resumen del documento

En este documento se recopila información sobre los requisitos que han requerido un análisis exhaustivo para su desarrollo.

En el análisis de cada requisito incluimos información sobre la descripción que se nos da del requisito, una conclusión tras analizarlo en la que se incluye las decisiones tomadas para llevar a cabo el requisito y, en algunos casos, un link a la validación del profesor sobre la decisión tomada.

4. Introducción

Un informe de análisis es un documento que recoge información sobre el proceso de análisis, en este caso, de los requisitos que han requerido dicho análisis

En ese análisis se plantean diversas formas de afrontar el problema que supone el requisito, las cuales se comparan para tomar una decisión final, decisión de la cual se debe explicar en qué consiste y por qué se ha elegido, teniendo en cuenta que esta decisión debe estar validada.

Para este documento se ha optado por usar una estructura clásica a la hora de realizarlo. Esta estructura está compuesta por una portada con información breve del grupo, así como un resumen del documento , una introducción, el contenido principal, una conclusión y la bibliografía consultada para su realización.

5. Contenidos

5.1. Requisitos de información

No ha sido necesario realizar un análisis de los requisitos de información debido a que este entregable no incorporaba requisitos de este tipo.

5.2. Requisitos funcionales

No ha sido necesario realizar un análisis de los requisitos funcionales debido a que los requisitos incluidos en esta categoría no ameritaban un análisis en detalle.

5.3. Requisitos no funcionales

REQUISITO 30): The system must show money amounts as they are entered by the users, but also their corresponding money exchanges according to the system currency if their known exchange ratios are recent. It is the students' responsibility to find the appropriate exchange-rate service; no implicit or explicit liabilities shall be covered by the University of Seville or their individual affiliates if the students hire pay-per-use services! This requirement must be fulfilled in this and every subsequent group or individual deliverable for it to be considered fulfilled.

CONTEXTO: Para este requisito, pensamos que donde radica la mayor dificultad es en establecer dónde implementar la funcionalidad.

Se plantean las dos siguientes alternativas para cumplimentar el requisito:

Alternativa 1: Implementar un método en cada servicio Show y List que cumpla este requisito. Dicho método se encargaría de tomar por un lado el tipo Money del formulario o de la lista, por otro lado la moneda configurada como la moneda del sistema y realizaría una conversión a ésta última.

Pros:

- No se aprecia ninguna ventaja para esta alternativa.

Contras:

- Se trata de una malísima práctica de reutilización de código.
- Reduce drásticamente la cohesión de los servicios implicados.

Alternativa 2: Crear una clase "MoneyExchange", la cual incluya el método anteriormente planteado y pueda llamarse en cualquier clase que se necesite.

Pros:

- Código mucho más legible.
- Buena práctica de reutilización de código.
- Soluciona el problema de la cohesión.

Contras:

- Podría suponer un ligero cambio en la estructura de ficheros del proyecto.

DECISIÓN TOMADA: Evaluando los pros y contras de ambas alternativas, es bastante obvio que la segunda alternativa es la opción más viable. Es por esto que elegimos crear la clase "MoneyExchange", y confirmarlo con el profesor en un Follow-up.

FUENTE: Follow-up.

REQUISITO 31): The system must prevent the principals from storing any data that can be considered spam. A piece of text is considered spam if the number of spam terms exceeds a predefined spam threshold. The default list of spam terms includes the following ones: “sex”, “viagra”, “cialis”, “one million”, “you’ve won”, “nigeria”, and their corresponding translations into the languages considered for internationalisation; the default spam threshold is 10%. Note that the previous default values can be changed at will by the administrators. Realise that a term must be considered spam irrespective of its case and the blanks in between its words; for instance, “one␣million” is a spam term that matches “one␣million”, “ONE␣MILLION”, “OnE␣␣␣MiLLiOn”, or “One␣⏎␣Million”; it doesn’t match “One␣Millionaire”, “One␣or␣two␣millions”, or “One␣sex␣million”, though. The spam detector must be reusable across different projects; that is: it must be implemented as an independent project that must be packaged into a reusable dependency. Do not forget to deliver your spam detector project so that it can also be evaluated or, otherwise, this requirement shall not be considered fulfilled. This requirement must be fulfilled in this and every subsequent group or individual deliverable.

CONTEXTO: Este requisito guarda una gran similitud con el anterior. Es por esto que ambos requisitos fueron analizados de la misma manera, se propusieron las mismas alternativas y se llegó a la misma conclusión para ambos.

Alternativa 1: Implementar un método en cada servicio Create, Update y Publish que cumpla este requisito. Dicho método se encargaría de tomar cada tipo String del formulario. Posteriormente, transformar dicho String a una cadena sin espacios ni caracteres y con todas las letras minúsculas. Por último, comprobaría si hay algún término de spam (definido previamente en una lista) incluido en el tipo String del formulario.

Pros:

- No se aprecia ninguna ventaja para esta alternativa.

Contras:

- Se trata de una malísima práctica de reutilización de código.
- Reduce drásticamente la cohesión de los servicios implicados.
- No sería exportable para otro proyecto.

Alternativa 2: Crear una clase “SpamDetector”, la cual incluya el método anteriormente planteado y pueda llamarse en cualquier clase que se necesite.

Pros:

- Código mucho más legible.
- Buena práctica de reutilización de código.
- Soluciona el problema de la cohesión.
- Sólo así sería exportable

Contras:

- Podría suponer un ligero cambio en la estructura de ficheros del proyecto.

DECISIÓN TOMADA: Al igual que en el requisito anterior, es muy evidente que la segunda alternativa es la mejor opción. De nuevo, consultamos con el profesor en un Follow-up y aprobó nuestra decisión.

FUENTE: Follow-up.

5.4. Requisitos de testing

No ha sido necesario realizar un análisis de los requisitos de testing debido a que el único requisito incluido en esta categoría no ameritaba un análisis en detalle del mismo.

5.5. Requisitos de gestión

No ha sido necesario realizar un análisis de los requisitos de gestión debido a que los requisitos incluidos en esta sección fueron analizados en entregas anteriores.

6. Conclusiones

Una vez finalizado este documento, con el qué damos por terminado el segundo informe de análisis del grupo C3.02.07, se han adquirido nuevos conocimientos sobre cómo listar, mostrar, editar, actualizar y eliminar entidades, conocimientos que nos servirán de ayuda en nuestro futuro en el mundo laboral.

Debido a conocimientos adquiridos previamente, varios requisitos han quedado fuera de este documento. Esto se debe a que gracias a estos conocimientos no ha sido necesario practicar un análisis sobre estos requisitos, ya que no mostraban demasiada complejidad. Entre estos requisitos se encuentran servicios de listar y mostrar detalles simples y sencillos de implementar; y documentos previamente elaborados en esta misma asignatura.

7. Bibliografía

- “On your tutorials” - Material proporcionado en la asignatura DP2 en la Universidad de Sevilla.