Informe de análisis sobre tarea 045

Integrantes del grupo C1.026:

* Ignacio Blanquero Blanco ([ignblabla@alum.us.es](mailto:ignblabla@alum.us.es))
* Adrián Cabello Martín ([adrcabmar@alum.us.es](mailto:adrcabmar@alum.us.es))
* María de la Salud Carrera Talaverón ([marcartal1@alum.us.es](mailto:marcartal1@alum.us.es))
* Joaquín González Ganfornina ([joagongan@alum.us.es](mailto:joagongan@alum.us.es))
* Natalia Olmo Villegas ([natolmvil@alum.us.es](mailto:natolmvil@alum.us.es))

Fecha: Sevilla, 6 de Marzo 2024

Tabla de contenido

[Resumen del Informe 3](#_Toc160639954)

[Historial de Versiones 4](#_Toc160639955)

[Introducción 5](#_Toc160639956)

[Informe de Análisis – Acme-SF-D02 6](#_Toc160639957)

[Registro de Análisis 1 6](#_Toc160639958)

[Requisito 6](#_Toc160639959)

[Análisis y decisiones 6](#_Toc160639960)

[Validación por el profesor 6](#_Toc160639961)

[Registro de Análisis 2 7](#_Toc160639962)

[Requisito 7](#_Toc160639963)

[Análisis y decisiones 7](#_Toc160639964)

[Validación por el profesor 7](#_Toc160639965)

[Registro de Análisis 3 8](#_Toc160639966)

[Requisito 8](#_Toc160639967)

[Análisis y decisiones 8](#_Toc160639968)

[Validación por el profesor 8](#_Toc160639969)

[Registro de Análisis 4 9](#_Toc160639970)

[Requisito 9](#_Toc160639971)

[Análisis y decisiones 9](#_Toc160639972)

[Validación por el profesor 9](#_Toc160639973)

[Registro de Análisis 5 10](#_Toc160639974)

[Requisito 10](#_Toc160639975)

[Análisis y decisiones 10](#_Toc160639976)

[Validación por el profesor 10](#_Toc160639977)

[Registro de Análisis 6 11](#_Toc160639978)

[Requisito 11](#_Toc160639979)

[Análisis y decisiones 11](#_Toc160639980)

[Validación por el profesor 11](#_Toc160639981)

[Registro de Análisis 7 12](#_Toc160639982)

[Requisito 12](#_Toc160639983)

[Análisis y decisiones 12](#_Toc160639984)

[Validación por el profesor 12](#_Toc160639985)

[Conclusiones 13](#_Toc160639986)

[Bibliografía 14](#_Toc160639987)

# Resumen del Informe

El objetivo de este informe es proporcionar detalles sobre el análisis de todos los tipos de requisitos especificados en esta segunda entrega del proyecto, así como las decisiones tomadas para llevarlos a cabo y enlazarlo, si procediera, a las validaciones y observaciones planteadas por el profesor encargado de la asignatura.

# Historial de Versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Versión** | **Contenidos** | **Fecha** | **Contribuyente** |
| V0.1 | Estructura inicial Analysis Report D02 | 21/02/2024 | Natalia Olmo Villegas |
| V0.2 | Actualizar Analysis Report D02 | 01/03/2024 | Natalia Olmo Villegas |
| V0.3 | Actualizar Analysis Report D02 | 05/03/2024 | Natalia Olmo Villegas |
| V0.4 | Actualizar Analysis Report D02 con comentarios del profesor de la segunda sesión de laboratorio | 06/03/2024 | Natalia Olmo Villegas |

# Introducción

En esta segunda entrega teníamos que implementar varios requisitos de información y un requisito de testing, a diferencia de la primera entrega, en la que sólo se tenía que implementar un requisito funcional. Por lo tanto, la carga de trabajo ha aumentado y resulta de vital importancia una buena gestión de las tareas a cumplimentar por parte del Project Manager, Joaquín González Ganfornina en este caso. A continuación, se describirán con más detalle todos los requisitos implementados individualmente por Natalia Olmo Villegas.

# Informe de Análisis – Acme-SF-D02

Cabe destacar que a la hora de declarar los requisitos he decidido que aquellos atributos en los que en su requisito correspondiente no se especificaba si era obligatorio se les asignara la etiqueta @NotNull para obligarlos a ser obligatorios.

## Registro de Análisis 1

### Requisito

Un contrato consiste en una o muchas reuniones entre los participantes (stakeholders) envueltos en el desarrollo del Proyecto. El Sistema debe almacenar los siguientes datos de ellos: un **código** (patrón “[A-Z]{1,3}-[0-9]{3}”, no puede estar en blanco, único), un **momento de instanciación** (en el pasado), un **nombre de un proveedor**, (no puede estar en blanco, debe contener menos de 76 caracteres), un **nombre de un cliente** (no puede estar en blanco, debe contener menos de 76 caracteres), algunos **objetivos** (no puede estar en blanco, debe contener menos de 101 caracteres), y un **presupuesto** (menor o igual que el coste correspondiente al proyecto)

### Análisis y decisiones

Dado que este requisito no presentaba mucha complejidad, no me ha surgido ningún problema, sólo se debían declarar los atributos siguiendo las instrucciones definidas en el requisito, las cuales eran bastante claras. Se ha creado un paquete “contracts” y dentro de este se ha creado la entidad “Contract.java”.

Lo único que no se ha podido implementar para esta entrega es la restricción de que el presupuesto fuera menor o igual que el coste correspondiente al proyecto, ya que para esta regla de negocio haría falta un servicio y la lógica de negocio de los servicios todavía no ha sido explicada en clase. Por lo tanto, se dejará para la siguiente entrega la implementación de esta funcionalidad, aunque se ha decidido que el atributo “budget” sea de tipo Money, un datatype que se encuentra en el framework proporcionado.

Además, se han establecido dos relaciones @ManyToOne, una de ellas hacia Project, ya que se especifica que un contrato serán un conjunto de muchas reuniones sobre el proyecto a tratar. La otra relación de este tipo es hacia “Client.java”, ya que se detalla que estas reuniones serán entre los stakeholders, haciendo referencia al rol cliente que se pedirá implementar más tarde.

### Validación por el profesor

El profesor revisó este requisito en la primera y segunda sesión de revisión de laboratorio y le pareció correcto.

## Registro de Análisis 2

### Requisito

Cada contrato tiene una evolución que está compuesta de registros progreso. El sistema debe almacenar la siguiente información sobre ellos: una identificación de registro (patrón “PG-[A-Z]{1,2}-[0-9]{4}”, no puede estar en blanco, único), un porcentaje de completitud (positivo), un comentario en el progreso (no puede estar en blanco, con menos de 101 caracteres), un momento de registro(en el pasado), y una persona responsable del registro (no puede estar en blanco, con menos de 76 caracteres)

### Análisis y decisiones

Dado que este requisito no presentaba mucha complejidad, no me ha surgido ningún problema, sólo se debían declarar los atributos siguiendo las instrucciones definidas en el requisito, las cuales eran bastante claras. Se ha creado un paquete “progresslogs” y dentro de este se ha creado la entidad “ProgressLog.java”.

Se ha decidido poner una restricción de máximo 100% para el grado de completitud, ya que un porcentaje no puede ser mayor que el 100%, para ello se ha usado la notación @Max y se ha introducido como parámetro 100, ya que esta restricción incluye en su rango el número dado como parámetro.

Por otro lado, se ha decidido establecer una relación @ManyToOne hacia la entidad “Contract.java” ya que se nos especifica que los contratos están compuestos de registros de progreso.

### Validación por el profesor

El profesor revisó este requisito en la primera y segunda sesión de revisión de laboratorio y especificó que sería necesaria la implementación de alguna restricción al atributo “completeness” para evitar que este grado de completitud superase el 100%.

## Registro de Análisis 3

### Requisito

El Sistema debe gestionar los paneles de los clientes con los siguientes datos: número total de registros de progreso con un rango de completitud por debajo del 25%, entre 25% y 50%, entre 50% y 75%, y por encima del 75%; media, desviación, mínimo y presupuesto máximo de los contratos.

### Análisis y decisiones

Dado que este requisito no presentaba mucha complejidad, no me ha surgido ningún problema. Se ha creado dentro del paquete forms una clase “ClientDashboard.java”.

Al comenzar la implementación de este requisito, se creía necesario la creación de un repositorio con sus correspondientes queries, pero el profesor indicó que esto no era necesario para esta entrega, ya que sólo se trataba de un requisito de información.

Por una parte, se ha decidido declarar los métodos correspondientes utilizando tipos primitivos ya que no aceptan valores nulos. Por otra parte, cabe destacar que debido a que la media se calcula realizando la suma de todos los componentes de un conjunto y dividiendo entre ellos, si se encontrara la situación en la que tenemos un conjunto vacío, no se debería permitir que se dividiera 0 entre 0, pues esto es infinito y podría dar diversos problemas. Entonces, para el caso de “averageBudgetOfContracts” se pedirá que devuelva un valor de tipo Double, ya que double no permite valores nulos y podría originar problemas en este caso.

Ocurre una situación parecida para todos aquellos métodos que están relacionados con los contratos, pues pueden generar un fallo en el sistema si se hicieran operaciones sobre un conjunto vacío, por lo que se ha decidido declarar para todos ellos que sean de tipo Double.

### Validación por el profesor

El profesor, tras revisar este requisito en la primera sesión de revisión del proyecto de laboratorio, indicó que deberíamos usar tipos primitivos como int y double, en lugar de Integer y Double para evitar que se recogieran resultados nulos. A su vez, nos indicó que para los métodos que implicaran calcular medias debíamos tener en cuenta que la media de un conjunto vacío podría dar problemas si no se declara el tipo adecuadamente.

Además, señaló que para esta entrega no era necesario y desarrollo de repositorios.

## Registro de Análisis 4

### Requisito

Producir datos de muestra variados para testear la aplicación informalmente. Los datos deben incluir 2 cuentas de cliente con las credenciales: “client1/client1” y “client2/client2”.

### Análisis y decisiones

Dado que este requisito no presentaba mucha complejidad, no me ha surgido ningún problema. He creado 1 archivo: “client.csv” y he modificado el archivo “user-account.csv”, ambos se encontraban en “*src/main/webapp/WEB-INF/resources/sample-data”.*

En el archivo “user-account.csv” se ha decidió a la hora de crear las nuevas credenciales, se ha decidido seguir la misma sintaxis al declarar key.

Tras haber completado ambos archivos, se probó a poblar la base de datos con mariaDB y Dbeaver y se lanzaron los launchers para conocer si se había completado la tarea con éxito y efectivamente los resultados fueron los esperados.

Para poblar las demás entidades se ha seguido la siguiente lógica: declarar una nueva fila para la tabla de la base de datos por cada prueba de los valores de los atributos, teniendo en cuenta que, para cada atributo, hay que probar 5 valores, uno en cada límite, otros próximos a los límites y otro en el medio. Esta tarea se ha desarrollado sin muchas dificultades creando nuevos archivos csv por cada entidad.

### Validación por el profesor

En este caso, por falta de tiempo en las sesiones de laboratorio, el profesor no ha podido revisar esta parte. No obstante, confiamos en que hemos desarrollado esta parte correctamente, pues al ver el registro de Dbeaver, comprobamos que se han rellenado las tablas correctamente.

## Registro de Análisis 5

### Requisito

Hay un nuevo rol específico del Proyecto llamado clientes, el cual tiene el siguiente perfil de datos: identificación (patrón “CLI-[0-9]{4}”, no puede estar en blanco, único), un nombre de compañía(no puede estar en blanco, con menos de 76 caracteres), tipo (no puede estar en blanco, puede ser “compañía” o “individual”), un correo electrónico (no puede estar en blanco) y un link opcional con más información.

### Análisis y decisiones

Dado que este requisito no presentaba mucha complejidad, no me ha surgido ningún problema, sólo se debían declarar los atributos siguiendo las instrucciones definidas en el requisito, las cuales eran bastante claras. Se ha creado una clase “Client.java” y un enumerado “TypeClient.java”, ambos dentro del paquetes “roles”.

### Validación por el profesor

El profesor revisó este requisito en la primera sesión de revisión de laboratorio y no encontró ningún problema.

## Registro de Análisis 6

### Requisito

Desarrollar un diagrama UML a partir de los requisitos individuales del estudiante 2.

### Análisis y decisiones

Para ello se ha usado la herramienta proporcionada por los profesores de clase UMLet y se han diseñado las restricciones establecidas en cada una de las entidades creadas. Estas relaciones se han explicado anteriormente. Tenemos dos relaciones unidireccionales @ManyToOne y una composición. Además, el formulario “ClientDashboard.java” se encuentra aislado, ya que estos formularios no se relacionan con nada en el UML.

Para la representación de estos requisitos se ha requerido añadir a este diagrama UML la entidad “Project.java”, debido a su relación con “Contract.java”.

Se ha decidido además establecer distintos colores en función de si lo que se estaba representado se trataba de una entidad, un rol, un formulario, un enumerado o una regla de negocio. Por último, también se ha llegado a la conclusión de que un contrato es decido por un grupo de clientes, stakeholders, de un mismo proyecto. Por simplicidad, se ha recogido que la clase “Client.java” hace referencia a ese grupo de clientes de un mismo proyecto. Y se ha decidido que sea una relación 1..\*-1 ya que para que el proyecto pueda avanzar debe haber un contrato al menos. No obstante, las otras relaciones las he diseñado con relaciones @ManyToOne 0..\*-1 ya que no se especifica un mínimo de 1 para que puedan coexistir.

### Validación por el profesor

El profesor no pudo revisar este requisito por falta de tiempo en la sesión de revisión de laboratorio.

# Conclusiones

En esta segunda entrega, nos hemos enfrentado a un aumento significativo de la carga de trabajo al implementar varios requisitos de información y un requisito de testing por parte de la tarea individual de cada uno de los miembros del equipo. La gestión efectiva de estas tareas ha sido dirigida por el Project Manager, que ha sido Joaquín González Ganfornina en esta ocasión.

La implementación de todos los requisitos asignados para esta entrega se ha llevado a cabo en general sin grandes complicaciones de manera eficiente.

Las decisiones más notables serían: la restricción de presupuesto respecto al coste del proyecto se pospuso para la próxima entrega, ya que requiere la comprensión de servicios y lógica de negocio aún no explicados, la decisión de utilizar tipos primitivos y la sugerencia del profesor de evitar el desarrollo de repositorios para esta fase.

En resumen, se ha cumplido con una exitosa implementación de todos los requisitos necesarios para esta entrega y los comentarios del profesor en las sesiones de revisión han sido positivos.

# Bibliografía

En blanco a propósito.