Informe de análisis grupal

Integrantes del grupo C2.026:

* Ignacio Blanquero Blanco ([ignblabla@alum.us.es](mailto:ignblabla@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))

Fecha: Sevilla, 8 de Julio 2024

Tabla de contenido

[Resumen del Informe 4](#_Toc170984750)

[Historial de Versiones 5](#_Toc170984751)

[Introducción 6](#_Toc170984752)

[Registro de Análisis 7](#_Toc170984753)

[Requisito 1 7](#_Toc170984754)

[Descripción 7](#_Toc170984755)

[Análisis y decisiones 7](#_Toc170984756)

[Validación por el profesor 8](#_Toc170984757)

[Requisito 2 9](#_Toc170984758)

[Descripción 9](#_Toc170984759)

[Análisis y decisiones 9](#_Toc170984760)

[Validación por el profesor 9](#_Toc170984761)

[Requisito 3 10](#_Toc170984762)

[Descripción 10](#_Toc170984763)

[Análisis y decisiones 10](#_Toc170984764)

[Validación por el profesor 10](#_Toc170984765)

[Requisito 4 11](#_Toc170984766)

[Descripción 11](#_Toc170984767)

[Análisis y decisiones 11](#_Toc170984768)

[Validación por el profesor 11](#_Toc170984769)

[Requisito 5 12](#_Toc170984770)

[Descripción 12](#_Toc170984771)

[Análisis y decisiones 12](#_Toc170984772)

[Validación del profesor 12](#_Toc170984773)

[Requisito 6 13](#_Toc170984774)

[Descripción 13](#_Toc170984775)

[Análisis y decisiones 13](#_Toc170984776)

[Validación del profesor 13](#_Toc170984777)

[Requisito 7 14](#_Toc170984778)

[Descripción 14](#_Toc170984779)

[Análisis y decisiones 14](#_Toc170984780)

[Validación por el profesor 14](#_Toc170984781)

[Requisito 8 15](#_Toc170984782)

[Descripción 15](#_Toc170984783)

[Análisis y decisiones 15](#_Toc170984784)

[Validación del profesor 15](#_Toc170984785)

[Requisito 9 16](#_Toc170984786)

[Descripción 16](#_Toc170984787)

[Análisis y decisiones 16](#_Toc170984788)

[Validación del profesor 16](#_Toc170984789)

[Conclusiones 17](#_Toc170984790)

[Bibliografía 18](#_Toc170984791)

# 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 | Actualizar Analysis Report D02 | 06/03/2024 | Natalia Olmo Villegas |
| V0.2 | Actualizar Analysis Report D02 | 08/03/2024 | Natalia Olmo Villegas  María de la Salud Carrera Talaverón  Ignacio Blanquero Blanco  Joaquín González Ganfornina  Adrián Cabello Martín |
| V1.0 | Actualizar requisitos tras segunda convocatoria | 04/07/2024 | María de la Salud Carrera Talaverón |

# 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 por todos los integrantes del grupo.

# Registro de Análisis

Cabe destacar que, a la hora de declarar los requisitos, aquellos atributos que no se especificara que debieran ser obligatorios se han asumido como tal.

## Requisito 1

### Descripción

Una notificación es un mensaje publicado por un director autenticado. El sistema debe almacenar los siguientes datos sobre ellas: el momento de instanciación (en el pasado), un título (no en blanco, de menos de 76 caracteres), un autor (no en blanco, de menos de 76 caracteres), un mensaje (no en blanco, de menos de 101 caracteres), una dirección de correo electrónico opcional y un enlace opcional. El autor debe computarse de la siguiente manera: “〈nombre de usuario〉 - 〈apellido, nombre〉”, donde “〈nombre de usuario〉” denota el nombre de usuario del autor que ha publicado la notificación y “〈apellido, nombre〉” denota su nombre completo.

### Análisis y decisiones

Para implementar este requisito, se ha creado una clase Java que extiende a la clase AbstractEntity de Acme Framework. La clase, Notice.java, tiene las anotaciones “@Entity” de Javax y “@Getter” y “@Setter” de Lombok. Se ha definido en el paquete “acme/entities/notices”. No tiene relación con ninguna otra entidad.

Imagen que contiene Tabla

Descripción generada automáticamenteLos atributos definidos son: instantiationMoment, de tipo Date, que no puede ser nulo y debe estar en el pasado; title, de tipo String, que no puede estar en blanco y tiene como máximo 75 caracteres; message, de tipo String, que no puede estar en blanco y tiene como máximo 100 caracteres; email, de tipo String, que debe estar formado como una dirección de correo electrónico; link, de tipo String, que debe estar formado como un enlace; username, de tipo String, que no puede estar en blanco; fullName, de tipo String, que no puede estar en blanco y debe seguir un patrón “cadena, cadena”. Tanto username como fullName se utilizan para definir al autor de la notificación, que se trata de un atributo derivado. Esto se indica utilizando la anotación “@Transient” de Javax.

Texto

Descripción generada automáticamenteEl autor se crea cogiendo el atributo username, añadiéndole el separador “ – “ y concatenándole al final el atributo fullName. La forma en la que se definen tanto el nombre de usuario como el nombre completo se verán en el requisito que implementa la funcionalidad de creación de una notice, al igual que la restricción de que el atributo author no puede estar en blanco y debe tener como máximo 75 caracteres.

### Validación por el profesor

Tras la entrega final de la primera convocatoria, este requisito no fue validado por el profesor por el siguiente motivo: “Corrección: El modelo considera que el campo ‘author’ es un campo persistente, cuando claramente se indica que se trata de un valor calculado. Se comprueba en la implementación que tampoco se hace el cálculo del atributo tal y como se indica en los requisitos.”

Para corregir el requisito, se añadieron los atributos username y fullName, se eliminó el atributo persistente author y se definió este atributo de nuevo como uno derivado. Hubo que modificar el archivo CSV de población para que se ajustara a la nueva implementación.

## Requisito 2

### Descripción

El Sistema de configuración debe incluir los siguientes datos iniciales:

* Un sistema monetario, que debe ser inicializado a “EUR”
* Una lista de las monedas aceptadas, las cuales deben ser inicializadas a “EUR”, “USD” y “GBP”

### Análisis y decisiones

Este requisito ha resultado un gran reto para María de la Salud Carrera Talaverón y Natalia Olmo Villegas, pues no encontrábamos demasiada información acerca de qué podíamos hacer para satisfacer dicho requisito.

En un primer momento, pensamos que sería buena idea crear un datatype basado en el datatype “Money.java” que se nos proporcionaba en el framework. No obstante, el profesor indicó que en esta entrega no era necesario la implementación de datatypes nuevos.

Entonces, pensamos dos alternativas, modificar “application.properties” o crear una entidad nueva. Finalmente, llegamos a la conclusión de que sería mejor crear una entidad nueva, debido a que se sabe que modificar este archivo “application.properties” podría generar problemas.

La entidad creada se llama “SystemConfiguration.java” y se encuentra dentro del paquete “systemConfigurations”. Sus atributos serán la moneda por defecto y la lista de monedas que pueden utilizarse. Para cumplir con el requisito por completo, creamos y nuevo archivo denominado “system-configuration.csv” poblándolo con los datos necesarios para satisfacer las restricciones.

### Validación por el profesor

En la última sesión de laboratorio antes de esta segunda entrega, el profesor indicó que no sería necesario un datatype nuevo. Sin embargo, tampoco nos proporcionó mucha más información acerca de cómo debíamos desarrollar esta tarea, por lo que decimos crear una nueva entidad para ello por las razones comentadas anteriormente.

Además, tras la entrega final de la segunda convocatoria este requisito fue validado.

## Requisito 3

### Descripción

Un banner permite a los administradores publicitas productos, servicios u organizaciones. El sistema debe almacenar la siguiente información sobre ellos: el momento de creación o actualización de la instancia (en el pasado), el período en el que se muestra (debe comenzar en cualquier momento tras la creación o actualización y durar al menos una semana), un enlace a una imagen almacenada externamente, un eslogan (no en blanco, 75 caracteres o menos) y un enlace a un documento web.

### Análisis y decisiones

Este requisito, implementado por María de la Salud Carrera Talaverón, no ha supuesto grandes complicaciones más allá del período de muestra del banner. Se ha creado el paquete “acme.banners” y dentro se ha añadido la entidad Banner.java.

Todos los atributos pedidos se han implementado con las restricciones solicitadas. En el caso del período, se ha decidido implementar mediante dos atributos no nulos: startDate y finishDate. Debido a que las restricciones para estos atributos eran personalizadas y no se pueden indicar mediante las anotaciones habituales disponibles en Java, se ha dejado su implementación para el siguiente entregable, donde se hará en la clase servicio adecuada. Se pensó añadir un atributo derivado llamado displayPeriod, que calculase el período de tiempo transcurrido entre startDate y finishDate, pero finalmente se descartó como sugerencia por parte del profesor.

Del mismo modo, inicialmente se relacionó la entidad con el rol Administrator.java, pero esto también fue descartado como parte de la validación del profesor, ya que todas las entidades a implementar en la parte grupal debían estar aisladas.

### Validación por el profesor

Habiendo incluido los cambios ya mencionados, el profesor validó esta implementación el día 6 de marzo.

Además, tras la entrega final de la segunda convocatoria este requisito fue validado.

## Requisito 4

### Descripción

Producir un diagrama UML.

### Análisis y decisiones

Imagen que contiene Diagrama

Descripción generada automáticamenteEsta tarea fue asignada a María de la Salud Carrera Talaverón y Natalia Olmo Villegas. Su desarrollo fue relativamente sencillo, puesto que, como ya se ha mencionado, las entidades a representar no estaban relacionadas entre ellas ni con ninguna otra. Se ha utilizado el programa UMLet como se indica en los requisitos de la asignatura. El resultado se puede encontrar en la carpeta “src/reports/Group” bajo el nombre de UML-Group.uxf.

### Validación por el profesor

No hizo falta que el profesor validase esta tarea debido a su sencillez. El único feedback aportado se deduce de las sugerencias hechas en la implementación de cada una de las clases, ya que este diagrama es un reflejo de estas.

## Requisito 5

### Descripción

Una reclamación es una oposición o contradicción publicada por cualquier persona que se hace a algo que se considera injusto. El sistema debe almacenar los siguientes datos sobre ellas: un código (patrón "C-[0-9]{4}"), no en blanco, único), un momento de instanciación (en el pasado), un encabezamiento (no en blanco, inferior a 76 caracteres), una descripción (no en blanco, inferior a 101 caracteres), el departamento al que va dirigida (no en blanco, inferior a 101 caracteres), una dirección de correo electrónico opcional y un enlace opcional.

### Análisis y decisiones

Este requisito fue asignado a Adrian Cabello Martin.

Dado que este requisito no presentaba una complejidad significativa, su implementación se llevó a cabo sin dificultades notables. Se realizó simplemente la declaración de los atributos conforme a las restricciones definidas en el requisito, las cuales estaban claramente especificadas y documentadas. Como resultado, se estableció un paquete llamado "claims", dentro del cual se procedió al desarrollo de la entidad "Claim.java".

Es importante destacar que los atributos "code", "heading", "description", "department", "emailAddress" y "link" fueron definidos como tipo String, y cada uno de ellos fue configurado con sus restricciones respectivas según lo requerido. Por otro lado, el atributo "instantiationMoment" se designó como tipo Date, lo que lo distingue del resto de los atributos y proporciona la capacidad de almacenar información temporal relevante.

Además, cabe mencionar que al tratarse de una entidad asociada a la parte grupal del proyecto, se ha tomado la decisión de no establecer ninguna relación con otras clases por el momento. Este enfoque se alinea con la estructura y las necesidades actuales del proyecto, y brinda claridad en la implementación y mantenimiento del código.

### Validación del profesor

Durante la última sesión práctica antes de la entrega, el profesor revisó y aprobó el requisito, haciendo hincapié en un detalle importante: no existía ninguna relación establecida con otras clases.

## Requisito 6

### Descripción

Un riesgo, positivo o negativo, es un hecho que tiene un impacto determinado en el desarrollo de un proyecto y debe ser considerado por el administrador. El sistema debe almacenar los siguientes datos sobre ellos: una referencia (patrón “R-[0-9]{3}”), no en blanco, único), una fecha de identificación (en el pasado), un impacto (número real positivo) , una probabilidad, un valor (resultado de la multiplicación del impacto y la probabilidad), una descripción (no en blanco, de menos de 101 caracteres) y un enlace opcional con más información.

### Análisis y decisiones

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteDado que este requisito no presentaba una complejidad significativa, su implementación se llevó a cabo sin dificultades notables. Se realizó simplemente la declaración de los atributos conforme a las restricciones definidas en el requisito, las cuales estaban claramente especificadas y documentadas. Como resultado, se estableció un paquete llamado "risk", dentro del cual se procedió al desarrollo de la entidad "Risk.java".

Los atributos "code", "description", y "link" fueron definidos como tipo String, y cada uno de ellos fue configurado con sus restricciones respectivas según lo requerido. Por otro lado, el atributo "identificationDate" se designó como tipo Date, lo que lo distingue del resto de los atributos y proporciona la capacidad de almacenar información temporal relevante.

Por último, tenemos a “impact”, “probability” y “value” de tipo “double”, este último es un atributo derivado con la anotación @Transient. Cada uno de ellos fue configurado con sus restricciones respectivas según lo requerido.

Además, cabe mencionar que, al tratarse de una entidad asociada a la parte grupal del proyecto, se ha tomado la decisión de no establecer ninguna relación con otras clases por el momento. Este enfoque se alinea con la estructura y las necesidades actuales del proyecto, y brinda claridad en la implementación y mantenimiento del código.

### Validación del profesor

Durante la última sesión práctica antes de la entrega, el profesor revisó y aprobó el requisito, haciendo hincapié en un detalle importante: no existía ninguna relación establecida con otras clases.

Tras revisar este requisito durante la segunda convocatoria, se realizaron algunas modificaciones en los atributos y la versión final es la que se presenta en este informe.

## Requisito 7

### Descripción

El sistema debe manejar tableros de administrador con los siguientes indicadores: número total de directores con cada rol; proporción de avisos con una dirección de correo electrónico y un enlace; proporciones de objetivos críticos y no críticos; promedio, mínimo, máximo y desviación estándar del valor de los riesgos; promedio, mínimo, máximo y desviación estándar del número de reclamaciones publicadas durante las últimas 10 semanas.

### Análisis y decisiones

Desarrollada por Adrian Cabello Martin.

Dado el nivel de complejidad relativamente bajo asociado a este requisito, su implementación se llevó a cabo sin mayores dificultades. En el paquete “forms”, se ha creado la clase “AdministratorDashboard.java” de acuerdo con las especificaciones requeridas.

En cuanto a la elección de los tipos de datos para los métodos correspondientes, se ha optado por utilizar tipos primitivos, dado que estos no admiten valores nulos, lo que simplifica su implementación y mejora la integridad de los datos.

Es importante señalar que, al realizar los cálculos como la media, relación, desviación ,máximo y mínimo, la presencia de un conjunto vacío podría generar problemas. Por lo tanto, para los métodos “ratioOfNoticesWithEmailAndLink”, “ratioCriticalObjectives”, “ratioNonCriticalObjectives”, “averageValueInRisks”, “deviationValueInRisks” , “minumumValueInRisks”, “maximumValueInRisks”, “averageNumberOfClaimPostedOver10weeks”, “deviationNumberOfClaimPostedOver10weeks”, “minumumNumberOfClaimPostedOver10weeks” y “maximumNumberOfClaimPostedOver10weeks” se ha determinado que devuelvan un valor de tipo Double o Integer. Esta elección se justifica en que admiten valores nulos, lo que puede prevenir complicaciones en situaciones donde se encuentre un conjunto vacío.

### Validación por el profesor

Después de la evaluación inicial del requisito durante la segunda sesión de revisión del proyecto de laboratorio, el profesor recomendó el uso de tipos primitivos como int y double en lugar de Integer y Double, con el fin de evitar la posibilidad de recoger resultados nulos. Asimismo, destacó la importancia de considerar adecuadamente el tipo de dato al calcular medias, ya que la media de un conjunto vacío podría ocasionar problemas si no se declara correctamente.

## Requisito 8

### Descripción

Un objetivo permite a un usuario autentificado definir una meta o fin hacia el cual las acciones de un proyecto en específico deben ser dirigidas. El sistema debe guardar la siguiente información sobre ellos: un momento de instanciación (en el pasado), un título (no en blanco, con menos de 76 caracteres), una descripción (no en blanco, con menos de 101 caracteres), una prioridad (“Low”, “Medium”, “High”), un estatus para indicar si es crítico o no, una duración (debe empezar en cualquier momento después del momento de instanciación), un enlace opcional con más información.

### Análisis y decisiones

Este requisito era bastante simple. Siguiendo la descripción del requisito, se creó una entidad “Objective.java” con los atributos definidos, aplicando las restricciones correspondientes. Esta entidad se guardó dentro del paquete “objectives”.

Los atributos "title", "description" y "link" fueron definidos como tipo String, restringiendo sus valores según la descripción del requisito. Por otro lado, el atributo "instantiationMoment" se designó como tipo Date y se aplicó una restricción para que sea una fecha en pasado. Para la duración, se crearon dos atributos: “startTimeDuration” y “finishTimeDuration”. De momento estos dos atributos no tienen ninguna restricción ya que el requisito no nos dice que las fechas tienen que estar en pasado. Para el atributo “priority”, se creó un enum llamado “Priority.java” con los valores correspondientes, que se almacena también en el paquete “objectives”. Por último, la entidad “status” se define con el tipo primitivo boolean.

Esta entidad no tenía ninguna relación con las otras entidades del proyecto.

### Validación del profesor

Tras la revisión de este requisito por parte del profesor, decidimos realizar varios cambios en la entidad. Primero, definimos el tipo “status” como un tipo boolean, ya que, en primera instancia, usamos un tipo Boolean con la anotación @NotNull, lo cual es redundante. El otro cambio realizado es quitar las relaciones con otras entidades, ya que, en principio, esta entidad estaba relacionada con Project y con UserAccount.

## Requisito 9

### Descripción

Genera datos iniciales para iniciar tu proyecto. Debe incluir una cuenta de administrador con las credenciales “administrator/administrator”. Genera datos para testear tus requisitos. Debe incluir dos cuentas de administradores con las credenciales “administrator1/administrator1” y “administrator2/administrator2”.

### Análisis y decisiones

La primera parte de este requisito ya estaba hecha, ya que en el archivo “user-account.csv” del paquete “src\main\webapp\WEB-INF\resources\initial-data”, estaba creada la cuenta con credenciales “administrator/administrator”.

Para incluir las cuentas con credenciales “administrator1/administrator1” y “administrator2/administrator2”, se modificó el archivo “user-account.csv” del paquete “src\main\webapp\WEB-INF\resources\sample-data”, añadiendo dos líneas para crear estas cuentas.

Finalmente, se generaron datos para probar todas las entidades que se requerían en el trabajo grupal. Esto se hizo creando archivos csv con el nombre “entidad.csv” en el paquete “src\main\webapp\WEB-INF\resources\sample-data”. Para cada entidad, se generaron 10 objetos de prueba. Para los atributos de todas las entidades, se prueban los valores máximos, valores cercanos al máximo, valores mínimos, valores cercanos al mínimo y varios valores intermedios. Además, para los atributos que pueden ser nulos, se comprueba lo que ocurre cuando tienen algún valor y cuando tienen valor null.

### Validación del profesor

Tras la entrega final de la primera convocatoria, el profesor no solicitó cambios en este requisito; por tanto, tras revisarlo en busca de errores y no encontrar ninguno, este requisito se da por validado.

# Conclusiones

En esta segunda entrega del proyecto, se ha afrontado el desafío de implementar múltiples requisitos de información y un requisito de testing, lo que ha supuesto un incremento significativo en la carga de trabajo en comparación con la primera entrega. En este contexto, la gestión eficiente de las tareas por parte del Project Manager, Joaquín González Ganfornina, ha sido de vital importancia.

Se ha llevado a cabo un análisis detallado de cada requisito implementado por los integrantes del grupo, resaltando las decisiones tomadas y las acciones realizadas durante el proceso de desarrollo. Se han abordado diversos desafíos, como la interpretación de los requisitos, la selección de enfoques de implementación adecuados y la validación por parte del profesor.

Se ha demostrado un compromiso con la calidad y la integridad del trabajo realizado, adoptando decisiones fundamentadas y adaptándose a los comentarios y sugerencias del profesor durante el proceso de validación. Se han establecido prácticas de colaboración efectivas entre los miembros del equipo, lo que ha permitido abordar los desafíos de manera eficiente y cumplir con los objetivos establecidos.

En resumen, esta segunda entrega ha sido un hito significativo en el desarrollo del proyecto, marcado por el esfuerzo conjunto, la dedicación y la capacidad de adaptación del equipo. Se espera que este enfoque riguroso y colaborativo continúe guiando el éxito del proyecto en las próximas etapas.

# Bibliografía

En blanco a propósito.