INFORME DE ANÁLISIS

GRUPO: C1.02.11

Francisco Javier de la Prada Prados (fraprapra1@alum.us.es)
Pablo Quindós de la Riva (pabquide@alum.us.es)
María José Ruiz Vázquez (marruivaz1@alum.us.es)
Juan Luis Ruano Muriedas (juaruamur@alum.us.es)
Santiago Zuleta de Reales Toro (santizuleta11@gmail.com)

Índice

Resumen ejecutivo

En este reporte se desarrollará toda la información referente al análisis de los requisitos más complejos de cada uno de los entregables.
Para ello, contaremos con una copia literal del requisito al que se refiere el registro y daremos las conclusiones detalladas del análisis y decisiones tomadas para subsanar dicho requisito, si se diera el caso, aportamos un enlace a la validación realizada por un profesor.

Historial de versiones

Fecha	Versión	Descripción de los cambios	Sprint
15/02/2023	1.0	Creación de los documentos para el “Deliverable 1” e inicialización del proyecto	1
20/02/2023	2.0	Realización de los documentos para el “Deliverable 2”	2

1. Introducción

En primer lugar, decir que en este documento se analizarán y explicarán cada uno de los requisitos abarcados por los entregables .
Los pasos a seguir para afrontar este análisis será leer detenidamente todos los requisitos del nuevo delivery, hecho esto, tendremos que distinguir entre dos tipos de requisitos:

Por una parte, los requisitos que por su simplicidad o bien por ser de los primeros en el comienzo de este proyecto no necesitaran un análisis previo.

Por otra parte, los requisitos de código que por su dificultad necesitaran de un análisis para su comprensión y valoración de cómo empezar a desarrollarlos. Es evidente que cuando vayamos avanzando en el proyecto los requisitos de las entregas empezarán a complicarse cada vez más.

El análisis exhaustivo de estos requisitos será muy importante ya que este nos dará una idea previa de lo que tendrá que hacer el desarrollador que le toque realizar esa tarea, viéndose beneficiado el tiempo como el coste de la misma.
Además, nos permitirá realizar una repartición más equilibrada de los requisitos ya que al conocer la dificultad, balanceamos la carga de trabajo para que unos compañeros no realicen más trabajos que otros.

En definitiva, creo que este documento nos ayudará a todos los integrantes del grupo tanto a estar en contexto de los requisitos más difíciles del sprint, como a conocer las conclusiones detalladas del análisis y las decisiones tomadas para subsanar el requisito.

2. Contenidos

En este punto encontraremos dos partes diferenciadas del requisito que será analizado:

- Una copia literal del requisito al que se refiere el registro.
- Conclusiones detalladas del análisis y decisiones tomadas para subsanar el requisito.

2.1. Listados registro de análisis

IF-09: Moneda aceptada (fraprapra1)

La configuración del sistema debe incluir los siguientes datos iniciales:

- Una moneda del sistema, que debe inicializarse en "EUR".
- Una lista de monedas aceptadas, que debe inicializarse en "EUR", "USD" y "GBP".

Análisis y toma de decisiones

Para esta tarea ha sido necesario el asesoramiento por parte de mi profesor de prácticas ya que no sabía cómo realizar esta tarea de forma satisfactoria y no encontraba ningun ejemplo parecido en las prácticas o sesiones de teoria.

Finalmente, la resolución fue que esta tarea debía resolverse modelando una entidad llamada “SystemConfiguration” o “Currency” en donde debía poner los atributos “systemCurrency” y “acceptedCurrencies” y posteriormente rellenar esos datos en el “sample-data” con las monedas aceptadas.

IF-10: Entidad Peep (fraprapra1)

Un "peep" es un mensaje publicado por cualquier persona. El sistema debe almacenar los siguientes datos sobre ellos: un momento de creación (en el pasado), un título (no vacío, más corto que 76 caracteres), un apodo (no vacío, más corto que 76 caracteres), un mensaje (no vacío, más corto que 101 caracteres), una dirección de correo electrónico opcional y un enlace opcional.

Análisis y toma de decisiones

Para realizar esta tarea debo seguir los pasos se han explicado en las clases de teoría, concretamente la **“Lesson 2-Data Model: Session 1-A foundation”**, donde se explica claramente cómo modelar una entidad en Java, usando además las restricciones que proporciona el framework para que todos los atributos de esta sigan las reglas detalladas en el documentos de requisitos grupales.

Además en el workspace proporcionado por los profesores vienen ejemplos prácticos de como modelar correctamente entidades siguiendo los pasos correctos, por lo que puedo revisar dichos ejemplos para asegurar que estoy haciendo el modelado de forma correcta.

IF-11: Entidad Boletín

Se ha creado la entidad Bulletin, que corresponde a un mensaje publicado por un administrador. El sistema debe almacenar los siguientes datos sobre ellos: Un momento de instanciación (en el pasado), un título (no en blanco, menos de 76 caracteres), un mensaje (no en blanco, menos de 101 caracteres), un indicador para indicar si es crítico o no, y un enlace opcional con más información.

Análisis y toma de decisiones:

Para llevar a cabo la tarea, se crea la clase de Java correspondiente estableciendo los siguientes tipos para sus atributos:

- Date: momento de instanciación. Se elige un tipo Date para que el dato haga referencia al momento exacto de creación del boletín, a modo de 'Timestamp'
- String: título. Establece un título para el boletín creado
- String: mensaje. Establece un mensaje en el que se incluye la información detallada del boletín en cuestión
- Boolean: es crítico. Un atributo el cual indica si el boletín es crítico (valor TRUE) o si no lo es (valor FALSE)
- String: link. Un enlace opcional para el boletín

Los atributos incluyen anotaciones en las cuales realizan validaciones simples. Las que se consideran complejas se realizarán en los formularios posteriormente.
Por último, también se ha creado el modelo en el diagrama UML.

IF-12: Entidad Offer

Un Offer es un registro en el que un administrador anuncia algo. El sistema debe almacenar los siguientes datos sobre ellos: un momento de instanciación (en el pasado), un encabezado (no en blanco, de menos de 76 caracteres), un resumen (no en blanco, de menos de 101 caracteres), un período de disponibilidad (al menos un día después de que se instancia la oferta y debe durar al menos una semana), un precio (positivo, posiblemente cero) y un enlace opcional con más información.

Análisis y toma de decisiones:

Para llevar a cabo la tarea, se crea la clase de Java correspondiente estableciendo los siguientes tipos para sus atributos:

- Date: momento de instanciación. Se elige un tipo Date para que el dato haga referencia al momento exacto de creación del boletín, a modo de 'Timestamp'
- String: encabezado. Establece un encabezado para la oferta creada.
- String: resumen. Establece un resumen en el que se incluye la información detallada del boletín en cuestión
- Period: periodo de disponibilidad. Establece un periodo de disponibilidad en el cual la oferta está disponible.
- Double: precio. Pone un precio a la oferta.
- String: link. Un enlace opcional para el boletín

Los atributos incluyen anotaciones en las cuales realizan validaciones simples. Las que se consideran complejas se realizarán en los formularios posteriormente.
Por último, también se ha creado el modelo en el diagrama UML.

IF-13: Entidad Nota (juaruamur)

Una nota es un mensaje que proporciona un miembro del profesorado registrado. El sistema debe almacenar la siguiente información sobre ellas: un instante de tiempo, un título (not blank, menos de 76 caracteres),un autor (not blank, menos de 76 caracteres), un mensaje (not blank, menos de 101 caracteres),una dirección de correo opcional, y un link opcional. El autor debe quedar como: “⟨username⟩ - ⟨surname, name⟩”, donde “⟨user-name⟩” denota el profesor que escribio la nota y “⟨surname, name⟩” denota su nombre completo.

Análisis y toma de decisiones

Las notas son una entidad en la que cada uno de sus parámetros se ponen diferentes restricciones. Me he basado en las diapositivas de la asignatura y en las entidades de prueba del acme jobs para esta tarea. @NotBlank y @Lenght(max = 76 o 101) para las restricciones de string según lo requerido y para la restricción del autor se ha usado @Pattern(regexp = "^\\(\\w+\\) - \\(\\w+,\\s\\w+\\)$"). Por último se ha seleccionado @URL para el link y la clase Date para la instancia de tiempo.

IF-14: Banner Entity (marruivaz1)

Un banner permite a los administradores anunciar productos, servicios u organizaciones. El sistema debe almacenar los siguientes datos sobre ellos: un momento de instanciación/actualización (en el pasado), un periodo de visualización (debe comenzar en cualquier momento después del momento de instanciación/actualización y debe durar al menos una semana), un enlace a una imagen que debe almacenarse en otro lugar, un eslogan (no en blanco, inferior a 76 caracteres) y un enlace a un documento web de destino.

Análisis y toma de decisiones

Para la realización de está tarea me he apoyado en el framework y el proyecto Acme-jobs.
La dificultad ha venido por el atributo periodo de visualización ya que este debe de aparecer desde el momento de instanciación o actualización del producto , servicio u organización hasta una semana después pero no he encontrado etiquetas spring boot que me ayuden a resolver esto. Ni tampoco he encontrado ejemplos en el proyecto que ya nos dan resultado los profesores, así que aprovecho las clases de seguimiento para preguntarle al profesor por esta duda.
Como se trata de un atributo que se usará en el dashboard lo más sencillo es crear dos atributos, uno de fecha de inicio y otro de fin para luego cuando se necesite calcular el periodo.

IF-15: Administrator DashBoard (juaruamur)

El sistema debe almacenar estadísticas de administrador con lo siguientes indicadores: número de miembros del profesorado con cada rol; ratio de mensajes con ambos email y link; ratio de anuncios críticos y no críticos; media, mínimo, máximo y desviación del presupuesto ofertado por moneda; mediana, mínimo, máximo y desviación del número de notas colgadas en las últimas 10 semanas.

Análisis y toma de decisiones:

Para esta tarea nos pedían la creación de la clase “Dashboard” la cual la mayor dificultad es planificar cómo van a ser las clases que hagan de contenedor de datos. Me he basado en las diapositivas de clase y a partir de ahí he planeado la clase del número de cada rol se ha elegido hacer un Map<String,Integer> siendo String el nombre del rol e Integer el número de ellos. Para las medidas según las monedas se ha elegido un Map<String,Double> donde String es el nombre de la moneda y Double la cantidad de la medida. Para los demás datos se ha elegido Double que se ajusta bien.

TR-16: Datos iniciales (Todos)

Se producen datos para cuentas de administrador con credenciales “administrador/administrador”. Para las dos cuentas creadas se utilizan las credenciales “administrator1/administrator1” y “administrator2/administrator2”. Además se incluyen datos para las entidades grupales “Bulletin”, “Banner”, “Offer”, “Note” y “Peep”.

Análisis y toma de decisiones:

Para llevar a cabo la tarea no se ha realizado un análisis por la naturaleza simple de la propia actividad. Simplemente se han incluido datos en los ficheros ‘csv’ correspondientes a cada entidad, teniendo que es necesario añadir también la clave de la entidad con la cual hay alguna relación (en este caso solo ocurre para el administrador).

MR-19: UML (Todos)

Elaborar un gráfico UML con las entidades del proyecto.

Análisis y toma de decisiones

Para realizar esta tarea debemos seguir los pasos se han explicado en las clases de teoría, concretamente la **“Lesson 2-Data Model: Session 1-A foundation” y “Lesson 2-Data Model: Session 2-Sample Data”**, donde se pueden ver en las diapositivas numerosos ejemplos de cómo realizar un modelado UML correctamente utilizando la herramienta que nos proporciona el framework para dicha tarea.

3. Conclusión

Conclusivamente este documento nos va servir de apoyo para contar cómo hemos realizado los requisitos más complejos de los sprint, y detallar qué decisiones importantes hemos tomado y como se han solventado los posibles errores.
Además nos ayudará a conocer como el resto de desarrolladores se organizan, analizan sus tareas y qué métodos usan para corregir sus errores. Sirviendo de referente a los demás miembros del grupo.

En esta segunda entrega ha habido diferentes cuestiones conforme al desarrollo de las entidades y la clase “dashboard” pero creemos que hemos aplacado los problemas con éxito y conseguido el resultado deseado.

Pero el acuerdo final, es que para las próximas entregas cada desarrollador después de la asignación y realización de sus tareas, tendrá que analizar aquellas tareas que le han resultado de mayor complejidad y contar paso a paso como ha realizado la tarea y con qué errores se ha encontrado.

4. Bibliografía

Intencionadamente en blanco.