


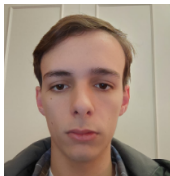


# Informe de análisis-javsanmar5



Santos Martín, Javier  
Ruíz Garrido, Javier  
Jiménez Morales, Francisco Miguel  
García de Tejada Delgado, José  
Antonio Daniel Porcar Aragón

# Información general del proyecto

PROJECT NAME			Acme-SF	
PARTICIPANTS				
Name	Email	Role	Username	Photo
Antonio Daniel Porcar Aragón	antporara@alum.us.es	Project Manager, Developer	antporara	
Francisco Miguel Jiménez Morales	frajimmor2@alum.us.es	Tester, Developer	frajimmor2	
Javier Santos Martín	javsanmar5@alum.us.es	Developer, Secretary	javsanmar5	
Javier Ruiz Garrido	javruigar2@alum.us.es	Analyst, Developer	Javiruizg	
José García de Tejada Delgado	josgardel8@alum.us.es	Operator, Developer	JoseGTD	
Stakeholders				
Francisco Miguel Jiménez Morales, Javier Ruiz Garrido, José García de Tejada Delgado, Javier Santos Martín, Antonio Daniel Porcar Aragón and José González Enriquez (the professor).				
Start date	Expected date of completion	Deliverables	Date of Document	
12/02/2024	27/05/2024	4	14/02/2024	

# Tabla de contenido

<b>Informe de análisis-javsanmar5</b>	<b>1</b>
<b>Información general del proyecto</b>	<b>2</b>
<b>Tabla de contenido</b>	<b>3</b>
<b>Resumen</b>	<b>4</b>
<b>Tabla de revisión</b>	<b>5</b>
<b>Introducción</b>	<b>6</b>
<b>Contenido</b>	<b>7</b>
<b>Conclusiones:</b>	<b>10</b>
<b>Bibliografía</b>	<b>11</b>

# Resumen

En este informe veremos un listado de los distintos requerimientos realizados por Javier Santos Martín, javsanmar5; aportando un análisis de aquellos que lo exijan.

Los requerimientos realizados son:

	MANDATORY	SUPPLEMENTARY
GRUPAL	<ul style="list-style-type: none"><li>Entidad objetivo</li></ul>	<ul style="list-style-type: none"><li>Diagrama UML</li></ul>
INDIVIDUAL	<ul style="list-style-type: none"><li>Entidad Sponsorship</li><li>Entidad Invoice</li><li>Dashboard sponsor</li><li>Datos de ejemplo Sponsor</li></ul>	<ul style="list-style-type: none"><li>Entidad Sponsor</li><li>Diagrama UML</li></ul>

Además de esto, se han modificado algunos aspectos generales del proyecto para alejarlo de la imagen de Acme SF-D01 y llegar a Acme SF.

Este documento se centrará en los cambios a nivel individual, dejando los requerimientos grupales para un posible «Analysis report» de grupo.

## Tabla de revisión

No procede.

# Introducción

Se ha realizado la creación de distintas entidades que se explicarán a continuación, un nuevo rol y un dashboard para este nuevo rol.

También se han modificado algunos datos de la imagen de marca.

La estructura del documento consta de los siguientes apartados:

- **Participantes:** Participantes del proyecto
- **Indice:** Indice del documento
- **Resumen:** Reducción del contenido
- **Introducción:** Breve introducción del documento
- **Contenido:** Contenido principal del reporte
- **Conclusión:** Cierre del tema

# Contenido

Lo aportado a nivel individual es lo siguiente:

## - CREACIÓN DE ENTIDADES:

**Sponsorship:** La creación de la entidad Sponsorship ha sido un proceso relativamente fluido, con excepción de un único obstáculo que ha surgido en relación con la propiedad "duration" y su implementación dentro del sistema. Esta entidad está estrechamente vinculada con las entidades Project e Invoice, tal como se detalla en el diagrama UML correspondiente.

En cuanto a las decisiones de diseño, se ha optado por abordar la variable "duration" mediante la inclusión de dos fechas: "startDate" y "endDate". Esta elección permitirá calcular la duración efectiva del patrocinio en el servicio, si bien dicho servicio aún no ha sido desarrollado.

Se han considerado otras alternativas durante el proceso de diseño. Entre ellas, la posibilidad de utilizar la clase Duration o la de implementar un nuevo DataType específico para esta propiedad. Sin embargo, la opción seleccionada se fundamenta en varios factores. En primer lugar, la clase Duration no ha sido abordada en el contexto de la asignatura, lo que implicaría una curva de aprendizaje adicional quizás innecesaria. Por otro lado, la creación de un nuevo DataType se vió como una medida excesivamente compleja y elaborada para una única propiedad.

En resumen, la decisión de emplear dos fechas para representar la duración del patrocinio fue la más sencilla y directa, evitando así complicaciones innecesarias en el desarrollo del proyecto. Este enfoque simplificado permitirá avanzar de manera eficiente en la implementación de Sponsorship, manteniendo la coherencia con el resto del sistema y minimizando posibles contratiempos en el proceso de desarrollo.

Archivos modificados:

- `.\src\main\java\acme\entities\sponsorship\Sponsorship.java`
- `.\src\main\java\acme\entities\sponsorship\SponsorshipType.java`

**Invoice:** La creación de la entidad Invoice también se ha realizado sin demasiadas complicaciones, aunque se han identificado algunos desafíos que han requerido una atención especial durante el proceso de desarrollo. Entre estos desafíos, se destacan los siguientes:

**Restricción para la propiedad dueDate:** Uno de los desafíos encontrados fue determinar dónde declarar la restricción para la propiedad "dueDate", la cual debía ser un mes posterior a la "registrationTime". Finalmente, se decidió implementar en el servicio (todavía no creado) puesto que no contabamos con restricciones ya creadas para algo así.

Cálculo del totalAmount: Otro desafío significativo fue el cálculo del "totalAmount", ya que este dependía de otras dos propiedades. El principal obstáculo radicaba en que la clase Money no contaba con un constructor que aceptara ambos valores (amount y currency) simultáneamente. Por lo tanto, fue necesario pensar otra forma de crear esta propiedad una vez completado el cálculo.

Ante este escenario, se consideraron diversas opciones para abordar el problema. Finalmente, se optó por crear una instancia vacía de la clase Money y asignar los valores correspondientes utilizando los métodos setAmount y setCurrency disponibles en dicha clase. Quedando de la siguiente forma:

```
@Transient
public Money getTotalAmount() {
    double taxToApply = (100.0 - this.tax) / 100.0;
    Double totalAmount = this.quantity.getAmount() * taxToApply;

    Money res = new Money();
    res.setAmount(totalAmount);
    res.setCurrency(this.quantity.getCurrency());

    return res;
}
```

En conclusión, si bien la implementación de la entidad Invoice ha presentado algunos desafíos, se ha logrado superarlos sin demasiadas complicaciones.

Archivos modificados:

- `.\src\main\java\acme\entities\invoice\Invoice.java`

**Sponsor:** La creación del rol Sponsor ha sido el proceso más fluido y sin contratiempos entre las tres entidades consideradas hasta el momento. Esta experiencia positiva puede atribuirse en gran medida a la familiaridad adquirida con el framework y a la ausencia de propiedades especialmente distintivas que requirieran un enfoque excepcional.

A diferencia de las entidades anteriores, el rol Sponsor no presentaba propiedades notablemente diferentes a las ya implementadas, lo que facilitó su integración en el sistema existente. Además, el equipo de desarrollo pudo capitalizar la experiencia previa acumulada durante el trabajo con el framework, lo que contribuyó a agilizar el proceso de creación y reducir la probabilidad de encontrar obstáculos significativos.

Archivos modificados:

- `.\src\main\java\acme\roles\Sponsor.java`

- **CREACIÓN DE FORMS:**



**SponsorDashboard:** La creación del SponsorDashboard implicaba el desarrollo de un panel de control específico para los sponsors, el cual proporcionaría acceso a diversos datos relacionados con las entidades previamente mencionadas. Para lograr este objetivo, se diseñó un Form que permitiera gestionar las propiedades requeridas en este contexto. Afortunadamente, el proceso de desarrollo transcurrió sin contratiempos ni imprevistos significativos.

Durante el proceso de diseño, se tomaron decisiones que influyeron en la implementación final del SponsorDashboard. Una de estas decisiones se relacionó con el uso de tipos primitivos en lugar de clases de Java, como `int/Integer`, para representar los datos. Aunque el uso de clases ofrece ciertas ventajas, como la capacidad de utilizar métodos asociados a estas clases, se optó por los tipos primitivos por varias razones.

En primer lugar, el uso de tipos primitivos elimina la posibilidad de que existan valores nulos, lo que simplifica el código ya que permite evitar la necesidad de agregar anotaciones y realizar comprobaciones activas del framework para garantizar la integridad de los datos.

En conclusión, la decisión de emplear tipos primitivos en lugar de clases de Java en el desarrollo del SponsorDashboard se basó en la necesidad de optimizar la eficiencia y la simplicidad del código, evitando posibles complicaciones asociadas con la gestión de valores nulos.

Archivos modificados:

- `.\src\main\java\acme\forms\SponsorDashboard.java`

## **-DATOS DE PRUEBA**

Se han diseñado casos de prueba para cada una de las clases mencionadas anteriormente, con el objetivo de garantizar su funcionamiento correcto y validar su comportamiento bajo diferentes escenarios. Estos casos de prueba se han documentado en ficheros CSV.

En la elaboración de estos casos de prueba, se ha prestado especial atención a la generación de datos límite que permitan evaluar los límites de la aplicación. Se ha procurado diseñar casos que abarquen una amplia gama de situaciones, centrándose en escenarios positivos y evitando la expectativa de errores. Este enfoque nos ha permitido evaluar la capacidad del sistema para manejar situaciones extremas y validar su robustez en condiciones diversas.

Es importante destacar que todos los casos de prueba diseñados se han ejecutado con éxito, lo que indica un buen comportamiento general de la aplicación. La superación satisfactoria de estos casos proporciona una sólida validación del sistema y nos da confianza en su capacidad para funcionar de manera fiable en entornos de producción.

Archivos modificados:

- `.\src\main\webapp\WEB-INF\resources\sample-data\sponsorship.csv`

- `.src\main\webapp\WEB-INF\resources\sample-data\invoice.csv`
- `.src\main\webapp\WEB-INF\resources\sample-data\sponsor.csv`

## Conclusiones:

En conclusión, el proyecto se desarrolló de manera efectiva, aunque surgieron algunos problemas durante el proceso de desarrollo, como la implementación de ciertas propiedades y la gestión de restricciones específicas en las entidades. Sin embargo, se pudo abordar estos desafíos y encontrar soluciones adecuadas, lo que garantizó el avance constante del proyecto sin demasiadas complicaciones.

También han existido algunas medidas que han requerido toma de decisiones, lo cual es apreciado ya que nos obliga a reflexionar sobre la mejor forma de abordar los problemas y nos permite tener creatividad en estos puntos en lugar de simplemente seguir procesos mecánicos.

Además, el uso de herramientas como Git y GitHub fue fundamental para el éxito del proyecto, ya que permitieron una colaboración fluida entre los miembros del equipo, facilitando la integración del trabajo individual y asegurando la coherencia y calidad del código en todo momento.

# Bibliografía

No procede.