03/03/2024

REPORTE DE ANÁLISIS

Grupo: C1.39

Repositorio: https://github.com/pabalcber/C1.039-Acme-SF

Integrantes:

Nombre	Apellidos	Correo Corporativo
Pablo	Alcántara Bernal	pabalcber@alum.us.es
María del Mar	Ávila Maqueda	maravimaq@alum.us.es
María	Barrancos Márquez	marbarmar16@alum.us.es
Sheng	Chen	sheche1@alum.us.es
Jun	Yao	junyao@alum.us.es

Tabla de versiones:

Fecha	Versión	Descripción de los cambios	Sprint
03/03/2024	1.0	Creación del documento y redacción de sus componentes	2
06/03/2024	1.1	Cambiado el análisis de varios requisitos	2
07/03/2024	1.2	Añadidos algunos análisis	2
07/03/2024	1.3	Revisión final	2

1.Índice

2. Resumen Ejecutivo

Este documento contiene el análisis de los requisitos referentes al entregable D02 del estudiante 2 perteneciente al grupo C1.39, incluyendo las descripciones tanto las tareas obligatorias como las suplementarias.

3.Tabla de revisiones

Número de revisión	Fecha	Descripción
1	07/03/2024	Revisión final antes de la entrega

4. Introducción

A continuación, se analizarán los requisitos 2,3,4 y 5 del D02 perteneciente al estudiante número 2.

5. Contenidos

Requisitos:

Requisito	Copia textual del requisito	Análisis y sus conclusiones
2	A **contract** is one or several agreements between the stakeholders involved in the development of a **project**. The system must store the following data about them: a **code** (pattern "[A-Z]{1,3}-[0-9]{3}", not blank, unique), an **instantiation moment** (in the past), a **provider name** (not blank, shorter than 76 characters), a **customer name** (not blank, shorter than 76 characters), some **goals** (not blank, shorter than 101 characters), and a **budget** (less than or equal to the corresponding project cost).	Tras revisar el requisito los únicos dos puntos no claros serían la relación que tiene un contrato con un proyecto, la relación entre contrato y progress log y la implementación de goals y budget. Implementación de goals: - **Entidad:** En este caso, un proyecto estaría compuesto por una o varias entidades externas de tipo goal. ○ **Pros:** a. Implementación más correcta. ○ **Contras:** a. Mayor dificultad de código. b. Aumenta el tiempo necesario para la implementación. - **Cadena:** En este caso, un proyecto tendría un String con los goals. ○ **Pros:** a. Facilita la implementación. b. Aprobada por el profesor. ○ **Contras:** a. No es la implementación más correcta. **SOLUCIÓN ELEGIDA:** Cadena. Implementación de budget: - **Mínimo en 0:** En este caso, un proyecto podría tener un contrato con mínimo presupuesto de 0. ○ **Pros:** a. Permite que haya proyectos cuyo coste sea nulo, como podría ser un proyecto de una ONG cuyo coste se paga con las donaciones recibidas. b. Proporciona un mayor rango ○ **Contras:** a. No es común este tipo de proyectos. - **Mínimo en 1:** En este caso, un proyecto podría tener un contrato con mínimo presupuesto de 1. ○ **Pros:** c. Proyectos más reales, ya que es poco común que haya proyectos de costo 0. ○ **Contras:**

b. Restringe más el rango de proyectos posibles a desarrollar.

SOLUCIÓN ELEGIDA: Mínimo 0.

Relación entre contrato y progress log:

- **Relación de agregación:** En este caso, un contrato podría tener una lista de progress logs asociados, pero los progress logs podrían existir independientemente de cualquier proyecto en particular.
  - **Pros:**
    a. Proporciona flexibilidad, ya que los progress logs pueden existir independientemente de cualquier contrato en particular.
    b. Da una mayor flexibilidad.
  - **Contras:**
    a. No tiene sentido que existan progress logs sin contratos asociados.
- **Relación de composición:** En este caso, un contrato tiene una lista de progress logs asociados, y los progress logs solo pueden existir si dependen de un proyecto en particular.
  - **Pros:**
    a. Especifica claramente que un progress log está asociado con un solo contrato, pero un contrato puede tener varios progress logs asociados.
    b. Tiene sentido que un progress log solo pueda existir en relación con un contrato.
  - **Contras:**
    a. Proporciona menor flexibilidad.

SOLUCIÓN ELEGIDA: Relación de composición.

Relación entre proyecto y contrato:

- **Relación ManyToOne:** En este caso, un proyecto estaría relacionado con uno o varios contratos, pudiendo existir proyectos sin contratos.
  - **Pros:**
    a. Pueden existir proyectos sin ningún contrato inicial asociado a los que se postulen varios clientes.
    b. Mayor flexibilidad.
  - **Contras:**
    a. La falta de contratos asociados puede dificultar el seguimiento y la gestión de los proyectos.
- **Relación de composición:** En este caso, un proyecto tiene uno o varios contratos asociados, sin que pueda existir un proyecto sin contrato.
  - **Pros:**

		a. Especifica claramente que un contrato está asociado con un solo proyecto, pero un proyecto puede tener varios contratos asociados. ○ **Contras:** a. Proporciona menor flexibilidad. b. No pueden existir proyectos sin ningún contrato inicial asociado a los que se postulen varios clientes **SOLUCIÓN ELEGIDA:** Relación ManyToOne.
3	Every **contract** has an evolution that is composed of **progress logs**. The system must store the following data about them: a **record id** (pattern "PG-[A-Z]{1,2}-[0-9]{4}", not blank, unique), a percentage of **completeness** (positive), a **comment** on the progress (not blank, shorter than 101 characters), a **registration moment** (in the past), and a **responsible person** for the registration (not blank, shorter than 76 characters).	Después de leer el requisito un par de veces no hubo grandes problemas para entenderlo, el único aspecto que puede llevar a dudas es la implementación de completenness. Implementación de completennes: - **double:** En este caso, completennes será un double que puede tomar valores entre 0.0 y 100.0. ○ **Pros:** a. Permite una mayor precisión en la representación de la completitud, ya que puede almacenar valores decimales entre 0.0 y 100.0. b. Ofrece una mayor flexibilidad para expresar niveles de completitud más precisos. ○ **Contras:** a. El uso de variables de tipo double puede llevar a problemas de redondeo y errores de precisión en cálculos matemáticos. - **int:** En este caso, completennes será un int que puede tomar valores entre 0 y 100. ○ **Pros:** a. Es más simple y directo, ya que los valores enteros entre 0 y 100 representan directamente el porcentaje de completitud, lo que facilita la interpretación y el uso de la información. ○ **Contras:** a. La limitación a valores enteros puede resultar en una menor precisión. b. Podría ser menos adecuado para proyectos que requieran una medición más detallada o que necesiten representar pequeños incrementos de completitud. **SOLUCIÓN ELEGIDA:** double.
4	The system must handle **client** dashboards with the following data: total number of **progress logs** with a	Este es el requisito que más problemas me ha causado. En primer lugar, surge la incertidumbre sobre si los atributos de clientDashboard deberían ser derivados u obtenidos directamente a través de consultas en el siguiente Sprint.

	completeness rate below 25%, between 25% and 50%, between 50% and 75%, and above 75%; average, deviation, minimum, and maximum **budget** of the **contracts**.	Si la segunda opción fuera la elegida, el clientDashboard no tendría relación con ninguna otra entidad ya que no persiste en la base de datos y por tanto no se puede realizar la relación, y simplemente se implementarían los atributos indicados en la descripción del requisito. Si consideramos el primer caso, surgirían varios problemas. El principal sería la necesidad de establecer una relación OneToOne entre cliente y clientDashboard. Además, aunque todos los atributos del clientDashboard son derivados y podrían calcularse a partir del cliente, sería necesario modificar las relaciones ya diseñadas en el UML. Esto nos deja con tres opciones: • **Bidireccionalidad:** Cambiar las relaciones entre cliente y contrato, y entre contrato y progress log a relaciones bidireccionales. Sin embargo, esta opción es desaconsejada debido a la complejidad que implica. ○ **Pros:** a. Permite mantener una estructura de relaciones más coherente y fácil de entender. ○ **Contras:** a. Implica una complejidad adicional en el diseño y la implementación. b. Usar relaciones no recomendades. • **Meter los datos manualmente:** Mantener las relaciones tal como estaban y agregar manualmente los datos del client dashboard. ○ **Pros:** a. Evita la modificación de las relaciones existentes y mantiene la simplicidad en el diseño. ○ **Contras:** a. Puede ser poco eficiente y propenso a inconsistencias en los datos. • **Cambiar Navegabilidad:** Mantener las relaciones tal como estaban, pero cambiar su navegabilidad para que el client dashboard pueda acceder al cliente y a los datos necesarios. ○ **Pros:** a. Permite mantener relaciones más simples y utilizar los datos existentes. ○ **Contras:** a. Se desvia de la navegabilidad recomendada. Tras discutirlo con el profesor, quedó claro que se implementaría como se describió en el segundo caso. Por otra parte, tenemos la implementación de average budget y deviation budget: - **Un único average y un único deviation:** En este caso, se implementaría un único atributo average y un único atributo

		deviation y para poder calcularlos habría que usar una API de conversión de tipos de monedas. ○ **Pros:** a. Se nos proporciona un ejemplo en Acme Jobs. b. Solución más correcta. c. Métodos más concis. ○ **Contras:** b. Más difícil de implementar. - **Un average y deviation por tipo de moneda admitido:** En este caso, se implementarían varios atributos average y deviation para que no se mezclen los distintos tipos de monedas usados. ○ **Pros:** a. No requiere el uso de una API por lo que es más sencillo de implementar. ○ **Contras:** a. Mayor cantidad de código. b. Código repetitivo. **SOLUCIÓN ELEGIDA:** un único average y un único deviation.
5	Produce assorted sample data to test your application informally. The data must include two **client** accounts with credentials "**client1**/**client1**" and "**client2**/**client2**".	Con respecto a este requisito, aunque inicialmente se nos solicita crear solo dos clientes, para poder comprobar todos los atributos es necesario generar más de dos clientes, lo que implica la creación de más de dos cuentas de cliente. • **Más de dos clientes:** Crear un número adicional de clientes y cuentas de cliente para garantizar la verificación completa de los atributos. ○ **Pros:** a. Permite una validación exhaustiva de todos los atributos y escenarios posibles. ○ **Contras:** a. Aumenta el tiempo de implementación. b. Los tests tardarán más en ejecutarse. • **Dos clientes:** Limitarse a crear solo dos clientes y cuentas de cliente, asumiendo que la prueba de estos casos será suficiente para garantizar la funcionalidad del sistema. ○ **Pros:** a. Reduce la complejidad y el tiempo requerido para crear y administrar las cuentas. b. Los tests tardarán menos tiempo en ejecutarse. ○ **Contras:** a. Puede dejar áreas de funcionalidad sin verificar, lo que podría resultar en problemas no detectados. **SOLUCIÓN ELEGIDA:** más de dos clientes.

Nota: Para todas las fechas implementadas en el pasado, se utiliza como fecha mínima el 01/01/2000, fecha recomendada por el profesor.

6.Conclusión

En esta segunda entrega, me he enfrentado a más requisitos ambiguos en comparación con la anterior. Aunque las ambigüedades no eran muchas ni muy problemáticas, han tenido un impacto significativo en el diseño de las nuevas funcionalidades. A pesar de esto, he logrado implementar con éxito todos los requisitos de esta entrega, tanto obligatorios como opcionales. Por lo tanto, considero que esta entrega ha sido exitosa.

7.Bibliografía

En blanco intencionalmente.