Calidad de software y gestión de deuda técnica

CSDT
Yudy Camila Fuentes Pardo
Mayo 2022
Escuela Colombiana de Ingeniería Julio Garavito

Resumen

Este documento presenta las mejoras y soluciones que nos puede ofrecer la implementación y gestión de calidad de software y deuda técnica, realizar la implementación en nuestro software y así ir revisando como esto puede beneficiar nuestros proyectos y si este es correctamente implementado, se realiza de manera consecutiva como puede mejorar la calidad de nuestro software y así ampliarlos como cultura organizacional y que esto sea implementado en todo momento.

Introducción

Definiremos en que consiste la calidad de software y deuda técnica para así conocer los beneficios y como implementarlo en nuestros proyectos y en cada etapa de nuestras soluciones de software.

Calidad de software: la calidad de un software es la manera de garantizar el éxito de un producto, su funcionalidad, mantenimiento, código limpio, esto para poder reducir los altos costos de pasos a producción, la inestabilidad del sistema y así pueda adaptarse y realizar mejoras en su desarrollo, haciendo que no sea complejo modificarlo.

Deuda técnica: esta deuda se presenta por realizar productos de software pensados en el corto plazo, es decir su construcción e implementación son pensadas para productos que no estarán tanto tiempo en el mercado, pero en caso de necesitarlos a largo plazo estos softwares necesitan cambios, su mantenimiento es costoso e imposible, por lo cual realizar cambios genera problemas en la construcción e incluso se identifican desarrollos que no son fáciles de mejorar o evolucionar.

La calidad de software y deuda técnica no es tenida en cuenta en la mayoría de los procesos debido a la urgencia del desarrollo, esto al que estimar tiempos siempre incrementa el desarrollo del software y toma más trabajo u esfuerzo, se admite que toda deuda es necesaria a revisar, pero esta nunca es tenida en cuenta antes del lanzamiento del producto.

Proyecto de investigación

Debido a que en el desarrollo tenemos varias ramas y especialidades, la deuda técnica tiene varias secciones las cual revisaremos en un listado en el cual nos enfocaremos y realizaremos mejoras, todo esto basado en un código Trivia Game.

Los tipos de deuda que tenemos son:

1. Architecture Debt
2. Build Debt
3. Code Debt
4. Defect Debt
5. Design Debt
6. Requirement Debt
7. Service Debt
8. Test Debt

9. Test Automation Debt
10. Documentation Debt
11. Infrastructure Debt
12. People Debt
13. Process Debt

Implementación deuda técnica

Realizaremos la implementación de deuda técnica y a lo largo del proyecto revisaremos e implementaremos las deudas técnicas para realizar las mejoras en nuestra trivia, esta información se puede ver detallada en la documentación en el repositorio: https://github.com/CamiFuentes17/trivia/blob/master/CSDT-2022-1.md, el código fuente fue tomado para la revisión y análisis de deuda técnica. [3]

En el desarrollo de nuestro proyecto empezamos a analizar nuestro código para así poder identificar y optimizar nuestro código fuente con esto revisar que tanto esto afecta nuestro código, encontramos con esta deuda que el mantenimiento del código era costoso esto se realizó mediante el análisis de code smells encontrando que el código tenia los siguientes:

- Dispensables: el código puede funcionar con la ausencia de este código.
- Duplicate Code: la duplicidad en nuestro proyecto existe por lo cual hace que este sea menos eficaz.
- Clases indispensables: sin esta clase el juego no funciona
- Acopladores: contribuyen a un acoplamiento excesivo entre clases o en su mismo código.[1]

Dando continuidad realizamos el proceso de refactorización en el cual pudimos encontrar varios por realizar, es importante tener en cuenta que los métodos excesivamente largos son la raíz de todos los males, por lo cual identificamos los siguientes:

1. Extract Method
2. Add Parameter
3. Rename Method
4. Rename Variable
5. Remove Dead Code
6. Substitute Algorithm
7. Change Reference to Value

Para progresar en la mejora de nuestro proyecto y mitigar la deuda técnica implementamos clean code y sus prácticas XP en el cual mejora notablemente la calidad de nuestro código de esta práctica encontramos:

1. El código no es entendible, ni testeable
2. El código cuenta con duplicidad
3. El código no es enfocado, cuenta con responsabilidades enfocadas a pocos métodos.
4. YAGNI (You Aren´t Gonna Need It): La trivia está generando código innecesario y no genera funcionalidad, no aporta o no es necesaria,
5. KISS (Keep it Simple, Stupid): El código puede ser simplificado y puede mejorar la calidad

En la revisión de cada una de estas deudas se reviso la necesidad de las pruebas y las afectaciones que puede tener en la arquitectura, por lo cual concluimos que:

- Deuda de pruebas: Al realizar pruebas unitarias estamos optimizando nuestro desarrollo y así mejorar la calidad del código, tener mejoras del equipo, aprender continuamente, recibir retroalimentaciones.

- Deuda de arquitectura: nuestro proyecto genera bastantes deudas de arquitecturas por su composición y falta de estructura, jerarquías, capas de desarrollo, el código es obsoleto, por lo cual su mantenimiento o reestructuración es demasiado costoso. [2]

Conclusiones

Para concluir tenemos que entender que el proceso de revisión de deuda técnica y calidad es indispensable en los desarrollos debido a que presentando estas mejoras podemos realizar integraciones continuas y tomar decisiones para implementar nuevas metodologías que nos permitan no afectar a las diferentes ramas, poder realizar priorizaciones en nuestros proyectos y realizarlos óptimos desde el inicio y no teniendo un retrabajo en el futuro.

Evidenciamos al implementar la deuda técnica y calidad en nuestros proyectos que podemos mitigar inconvenientes a largo plazo y ver la optimización de los sitios en nuestros desarrollos, haciéndolo más eficientes, y generando conocimiento que sea útil para todo el equipo, genere conocimiento y una cultura.

Referencias

[1] Alves, N. S., Ribeiro, L. F., Caires, V., Mendes, T. S., & Spínola, R. O. (2014, September). Towards an ontology of terms on technical debt. In *2014 Sixth International Workshop on Managing Technical Debt* (pp. 1-7). IEEE.

[2] Belingueres, G. (2017). *La gestión de la deuda técnica en los sistemas de información* (Doctoral dissertation, Universidad de Buenos Aires. Facultad de Ciencias Económicas.).
Sierra, G., Shihab, E., & Kamei, Y. (2019). A survey of self-admitted technical debt. *Journal of Systems and Software*, *152*, 70-82.

[3] GitHub, Trivia, J. B. Rainsberger , 2012
https://github.com/jbrains/trivia