UNIVERSIDAD NACIONAL DE SAN ANTONIO ABAD DEL CUSCO
FACULTAD DE INGENIERÍA ELÉCTRICA, ELECTRÓNICA, INFORMÁTICA Y MECÁNICA
ESCUELA PROFESIONAL DE INGENIERÍA INFORMÁTICA Y DE SISTEMAS

"ANÁLISIS DE CODE SMELLS"

DOCENTE: Ing. ROXANA LISETTE QUINTANILLA PORTUGAL

CURSO: INGENIERÍA DE SOFTWARE

ESTUDIANTES: CÓDIGO:

- BUSTAMANTE FLORES ERICK ANDREW 171943 P
- CHOQUE SARMIENTO LEYDI DIANA 174909 P
- HUANCARA CCOLQQUE ALEX HELDER 174911 P
- INCA CRUZ CARLOS EDUARDO 174912 P
- PEREIRA CHINCHERO RICHARD MIKHAEL 171916 P
- QUISPE CHAMBIA CARLOS ENRIQUE 174447 P
- SARCO JACINTO DANIEL EDUARDO 174452 P
- VEGACENTENO OLIVERA RONALDINHO 140934 P

P = Participó
NP = No Participó

CUSCO - PERÚ
2021

1. **CODE SMELLS - DEFINICIÓN**

    - Un Code Smell son una serie de síntomas en el código que nos vienen a indicar que tal vez no se están haciendo las cosas de una forma del todo correcta.

    - El no tener un control correcto de esta serie de códigos puede llevar a que haya algún problema a futuro.

    - También son conocidos como un código que huele o apesta.

2. **CODE SMELLS - CARACTERÍSTICAS**

    - El código limpio es obvio para otros programadores.

    - El código limpio no contiene duplicaciones.

    - El código limpio contiene un número mínimo de clases y otras partes móviles.

    - El código limpio pasa todas las pruebas.

    - El código limpio es más fácil y económico de mantener.

3. **TIPOS DE CODE SMELLS EN EL PROYECTO**

    Existen varios tipos de Code Smells, desde abuso de Programación orientada a Objetos(POO), Códigos dispersado, etc. Este análisis corresponde a los tipos de Code Smells que presenta nuestro proyecto. De los cuales son los siguientes:

    DISPENSABLE: Un dispensable es algo inútil e innecesario cuya ausencia haría el código más limpio, más eficiente y más fácil de entender. En nuestro proyecto se tiene los siguientes tipos de códigos Dispensables:

    1. *"Código repetido en los cruds, en front y en el back".*

    2. *"Clase que solo tiene una función y un solo llamado por un componente y no es dinámico".*

    BLOATERS: Son códigos, métodos y clases que han aumentado a tales proporciones que es difícil trabajar con ellos. En nuestro proyecto se presentan los siguientes tipo de código Bloaters:

    1. *"Importamos todas las funciones que nos ofrece una librería pero no usamos todas, además quedan variables de las funciones declaradas sin usar".*

    ACOPLADORES: Son códigos, métodos y clases que contribuyen a un acoplamiento excesivo, es decir abuso de funciones o clases que sobrecargan el programa. En nuestro proyecto se presentan el siguiente código del tipo Acoplador:

1. *"Los Códigos funcionales .js tienen partes de codificación para el diseño y no todo está en un archivo separado .css".*

A continuación se muestran todos los tipos de código con respecto a los comentarios

Nro	Lista de Code Smells	Tipo de Code Smells
1	Código repetido de los CRUDs en Front-End y el Back-End	DISPENSABLE
2	Clase que solo tiene una funcion y solo es llamado por un componete, y no es dinamico	DISPENSABLE
3	Importamos muchas funciones pero no usamos todas, ademas quedan variables de las funciones declaras sin usar.	BLOATERS
4	Los Códigos funcionales .js tienen partes de codificación para el diseño y no todo esta en un archivo separado .css	ACOPLADORES

de los desarrolladores.

4. ¿POR QUÉ HEMOS CAÍDO EN LO CODE SMELLS?

1	La Falta de experiencia de los integrantes con el lenguaje de programacion que nos lleva a guiarnos con otros trabajos que pueden tienen code smells y el no utilizar correctamente las librerias.
2	El trabajo individual de cada integrante al hacer el la division no siempre se adecua a la forma de trabajo o de programacion de los demas, lo que ocasiona dificultad en la integración y produce code smells.

5. MEJORAS EN EL PROYECTO

Nro	Lista de Code Smells	Corrección

1	*"Código repetido en los cruds, en front y en el back".*	Crear una clase padre con las principales funciones para los CRUDs
2	*"Clase que solo tiene una función y un solo llamado por un componente y no es dinámico".*	Añadir esta clase al componente de la cual es llamado para que sea un solo archivo general.
3	*"Importamos todas las funciones que nos ofrece una librería pero no usamos todas, además quedan variables de las funciones declaradas sin usar".*	Identificar y Eliminar las variables que no se usan
4	*"Los Códigos funcionales .js tienen partes de codificación para el diseño y no todo está en un archivo separado .css".*	Extraer los códigos relacionados al diseño de las interfaces del código funcional y colocarlos todos en archivos .css

6. CONCLUSIONES

- Gracias a la última clase del presente curso el equipo de desarrollo tuvo conocimiento de algunos posibles errores que se convertirían en problemas a futuro.

- El conocimiento de los Code Smells resultó ser algo nuevo que todos en el equipo de desarrollo no conocían, este reciente conocimiento nos permitirá tener en cuenta algunos de estos tipos de Smells en el proyecto a futuro.

- Entre nuestras consideraciones de Code Smells a Futuro vendrían en la parte del diseño de las demás interfaces y también en la implementación de funciones adicionales, problemas que se deben tener un control.

7. BIBLIOGRAFÍA

- **https://refactoring.guru/refactoring/smells**

- **https://apiumhub.com/tech-blog-barcelona/code-smells/**

- **https://openwebinars.net/blog/code-smells-y-deuda-tecnica/**