# Introducción

El objetivo de todo esto, del Clean Code, de lo que vamos a ver Refactoring de tantas otras técnicas y prácticas, tienen que ver con que nosotros podamos entenderlos

Ya que usualmente pasamos más tiempo leyendo el código que escribiéndolo y por eso es muy importante la legibilidad, mantenibilidad, etc ya que ahorran bastante tiempo y dinero.

# Definición

Es una técnica disciplinada que nos permite mejorar la estructura interna del código sin alterar su comportamiento.

Alteramos la estructura interna para mejorarla, par hacer que este código sea más fácil de entender y modificar, pero sin que esto afecte en ningún momento el funcionamiento observable de la aplicación se vea afectado. Son solo cambios internos y no cambios percibibles por el usuario.

Entonces se preguntarán refactoring es “¿limpiar el código?”, si es que olvidamos una de las partes más importantes de la definición podríamos decir que sí, pero refactoring no es solo limpiar el código, sino es una técnica DISCIPLINADA para limpiar el código de manera eficiente y controlada.

Asimismo el ser una técnica disciplinada significa que puede ser mejorada con la práctica y el tiempo, de tal manera que la apliques más eficazmente, con menos probabilidad de error y un mayor conocimiento de todas las estratégicas que existen.

# Porqué es importante

Porque necesito yo modificar algo que no está roto, si supuestamente está bien.

Cualquier desarrollador sabe que escribir código limpia a la primera es muy difícil; aun así consiguiéramos hacer esto, este código se va a degradar de manera inevitable con el tiempo, debido a los constantes cambios y modificaciones.

Lo que significa que el riesgo de que nuestro código se vuelva poco mantenible con el tiempo siempre va a existir y si no tomamos ninguna acción este riesgo se va a transformar en la pérdida de capacidad de dar valor de manera continua y constante al cliente, que es cuando los proyectos fracasan o las aplicaciones mueren y son cambiadas.

Muchos de las veces pueden ser preventivos (reducir bugs, aumentar la reusabilidad y mantenibilidad), u otros estratégicos (nueva funcionalidad) y otros como: al arreglar un bug, al realizar revisión de código.

# Refactoring Catalog

Cuando limpiamos el código, esto lo hacemos aplicando una serie de pequeñas transformaciones a este código, cada una de estas transformaciones es llamado un refactoring.

Debido a que el cambio, que involucra varios de estos refáctorings, puede ser fácilmente identificado se ha podido nombrar clasificar y definir varios de ellos.

Estas transformaciones son cambios bien pequeños a la estructura del código pero aplicar consecutivamente una secuencia o grupo de estas se puede lograr un cambio bien significativo.

# Herramientas.

Mover métodos de un lado, descomponerlos, etc es una tarea que toma tiempo y muy propensa a errores si la realizamos de forma manual.

La buena noticia es la gran mayoría de IDES ya proveen de muchas herramientas para realizar estas refactorizaciones de manera automática y sin mucho esfuerzo.

Incluso estas herramientas respetan la misma categoría, nombres y procedimiento que hablábamos anteriormente, por lo tanto conocer, entender claramente y si es posible aplicar estas refactorizaciones de manera manual es esencial antes de poder utilizar las herramientas automatizadas.

# Code Smells

Ahora que ya tenemos una idea de qué es refactoring. Pero decidir cuando comenzar y cuando detenernos es algo tan fundamental como conocer las mecánicas del refactoring.

Explicar cómo limpiar el código (crear métodos, eliminar variables) es relativamente fácil ya que son procedimientos de cierta manera definidos sin mucha ambigüedad. Pero explicar cuando se deben hacer estos cambios podría estar sujeto a una noción un poco vaga de estética en el código. Entonces lo que necesitamos son algunos criterios un poco más solidos.

Aquí es cuando entra el término code smells, que no es otra cosa que criterios más solidos o señales de advertencia que podemos percibir en nuestro código de cuando probablemente sería una buena idea aplicar el refactoring.

# Code Smells Catalog

Ciertamente no son criterios precisos, ya que no existe ninguna métrica que sea rival ante la intuición y experiencia humana, pero son buenos indicadores a tomar en cuenta de que probablemente deberíamos tomar una acción en el código. Lo que también significa que debemos desarrollar ese sentido o intuición de cuando muchas líneas de código son muchas y cuando muchas variables son realmente muchas.