Primeros pasos

José Luis Mena Jiménez

Octubre 2022.

Red5G.

SCRUM

Es una metodología de trabajo en equipo para la creación e implementación de un proyecto, el cual determina el rol de cada integrante del equipo y la distribución de los recursos y el tiempo necesarios para el proyecto, con el fin de administrar y gestionar por medio de diferentes herramientas.

Scrum Daily: Son reuniones diarias las cuales se desea que no tome más de 15 minutos de tiempo, en donde cada integrante de equipo, expone las actividades que realizará en el día y los probable contratiempos que pueda presentar, y presente un informe del estado de las actividades del último Daily.

Sprints: Son intervalos de tiempo para llevar a cabo actividades, este proceso se realiza múltiples veces en forma de ciclos, cumple la función de particionar el proyecto de tal forma que se determina un plazo de tiempo para hacer N número de actividades, y así avanzar paulatinamente el proyecto, algo como mini proyectos dentro del proyecto, que permite medir todo tipo de proceso y definir los puntos de inflexión en relación a lo no entregado o inconvenientes presentes durante el scrum.

Planificación del Sprint: Un segmento en el cual se determina las actividades que se realizarán en el sprint, tomado en cuenta al equipo de trabajo, de igual forma se toma en cuenta la entrega y el tiempo estipulado para ello y los posibles contratiempos.

Scrum Master: Es el que toma las riendas del manejo del equipo, quien da el enfoque al equipo sobre la metodología y de esta forma dirigir al equipo, hacer seguimiento de las tareas sobre los Sprints.

Product Owner: El que lleva el rumbo del proyecto, es decir toma la decisiones sobre requerimientos y resultados, quién evalúa según su criterio de aceptación, al culminar el proyecto por parte del equipo, el Product Owner, hace las respectiva revisión con los criterios de aceptación. De esta forma se determina sí cumple con todos los requerimiento planteado por parte del cliente, y hacer la debida corrección o entrega.

Código Limpio

Similar a una pintura o a una escultura, el código es como una obra de arte, solo que en lugar de ser una expresión de algún sentimiento, causa o alusión metafórica, debe parpar un propósito o función, lo cual se interpreta como una acción que cumple un objetivo. Así como la alquimia requiere de una habilidad de pensamiento lógico y precisión, además de una serie de puntos a tener en cuenta en el desarrollo desde las variables, la composición de las funciones y condicionales, lo anterior son variante que influyen para el entendimiento de código no sólo para el autor su autor sino para todo aquel que trabaje en el.

Los beneficios de un código limpio, facilita el hacer mantenimiento, agilizar el tiempo de desarrollo, disminuir la redundancia en el código y determinar que los tiempos de respuesta sean óptimos. Todo esto va de la mano con una excelente compresión de lo que está escrito y para esto es necesario un buen lector, al igual que un buen escritor.

Uno de los puntos que tenemos son los nombres de las variables, cada una es creada un con objetivo, que se extiende a lo largo de la vida del desarrollo o en un pequeño fragmento de este, independientemente del tiempo de vida es importante darle un nombre adecuado y conciso, el cual de paso ayude a determinar el propósito de la acción de la cual hace parte, y por consiguiente ayude a entender el léxico del código.

Teniendo en cuenta una lista la cual siempre va a tener números en ella, se busca determinar el promedio de esta. A Continuación hay dos códigos que están escritos de la misma forma con excepción del nombre de las variables.

Código 1

```php
<?php
    $l = [1, 2, 3, 4, 5, 6, 7, 8, 9];
    $a = 0;
    $b = 0;
    $c = 0;

    for ($i=0; $i < count($l); $i++) {
        $a = $a + 1;
        $b = $b + $l[$i];
    }

    $c = $b / $a;

    echo $c;
?>
```

Código 2

```php
<?php
    $lista_numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9];
    $contador = 0;
    $acumulador = 0;
    $promedio = 0;

    for ($i=0; $i < count($lista_numeros); $i++) {
        $contador = $contador + 1;
        $acumulador = $acumulador + $lista_numeros[$i];
    }

    $promedio = $acumulador / $contador;

    echo $promedio;
?>
```

Sin tener en cuenta el enunciado del problema, ¿Cuál de los dos códigos se entiende mejor? En un caso posterior, se solicitará a una persona diferente al autor modificar el código para hacer una operación con el valor de la sumatoria de cada número de la lista o la cantidad de número que hay en esta, En el caso del Código 1, inicialmente no se puede hacer una identificación a simple vista, es necesario hacer un análisis completo del código y determinar qué rol cumple cada variable, lo que sucumbiria un gasto de tiempo.
En el caso del Código 2, es más fácil deducir a simple vista la composición y función de este.

Puntos a tener en cuenta y errores cometidos

En el transcurso del desarrollo existen muchas formas en las que se pueden cometer errores, como código, funcionalidad y utilidad, formato, codificación o sintaxis. A pesar de que el compilador pueda ejecutar el software con éxito esto no quiere decir que se encuentre exento de errores, ya que en el tiempo de ejecución puede dar lugar a errores, que en teoría debían estar sufragados, tales como variables indefinidas, funciones mal ejecutadas, servicios en TimeOut, sin embargo esto se debe a varios factores entre lo cual se tiene en cuenta.

Requerimientos no claros por parte del desarrollador: En múltiples ocasiones el desarrollador puede dar por sentado un requerimiento y dejarlo a su propia interpretación lo cual en el momento, para proceder con la entrega de este. Pero esta práctica pone en riesgo el tiempo de desarrollo al no tener una lógica objetiva de este, y puede repercutir no solo en el requerimiento en sí, sino en el proyecto final. La causa de este factor se debe a que el desarrollador, omite hacer las preguntas pertinentes y aclara dudas sobre el requerimiento, sea por cualquier motivo, dejándolo con lo que sabe y llenado los huecos por su propia cuenta y conocimiento.

Interrupción del aprendizaje: Al ser el desarrollo una disciplina la cual se va actualizando constantemente, es necesario estar al día en esos conocimientos, no necesariamente a la par, sin embargo se requiere tener un conocimiento de las tecnologías recientes, en un intervalo no mayor a 3 años, debido a que pueden estar obsoletos, lo que daría como resultado que acciones simples, tome el doble o hasta el triple de tiempo de desarrollo con una tecnología en desuso que con una reciente.
Una tecnología reciente, la cual cuenta con herramientas que ayudan a simplificar el trabajo, ya que estas nacen como una solución a distintos problemas de sus predecesoras.

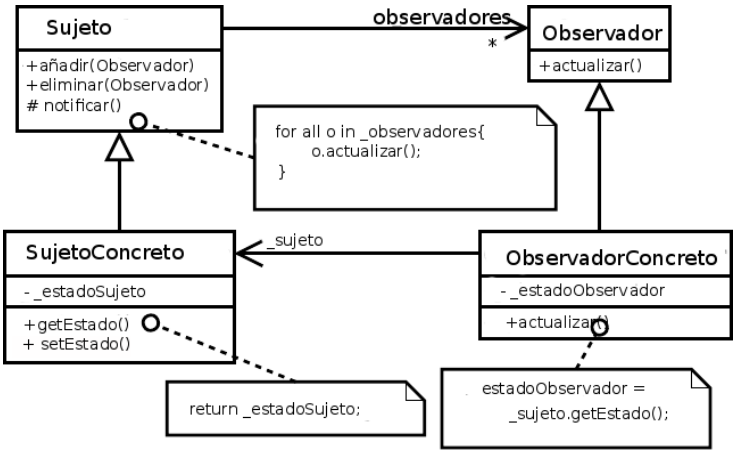
Apresurar trabajo: El hecho de terminar rápidamente una actividad que en un momento es compleja no es sinónimo de que esté bien hecha, puede significar que requiera más tiempo del que se planteó inicialmente, lo que la lleva a reestructurarse o empezar de nuevo. Omitir o ignorar los inconvenientes probables puede ser la razón principal de este caso, ya que sí no se mencionan no existen, pero el hecho de que no se vea no quiere decir que no exista. Es recomendable hacer un planteamiento de las actividades, teniendo en cuenta el tiempo adecuado y los posibles contratiempos que se puedan presentar para responder de la mejor forma posible y disminuir cualquier coste adicional sobre el proceso.

Gestión del tiempo: Un punto promesa que al inicio y durante todo el desarrollo es importante es el tiempo. Planificar el tiempo que llevará desarrollo de un proyecto como sus tareas y/o actividades son unos los puntos más relevantes, ya que permite tanto al equipo como a sus integrantes hacer una manejo y seguimiento de su tiempo, permitiendo desempeñar la distinta actividades, y a su vez las consecuencias de estas.
Para una materia la cual requiere un esfuerzo mental, es vital tener objetividad y lógica. Tener inteligencia emocional ayuda con las interacciones y toma de decisiones durante el desarrollo.

Patrones de diseño

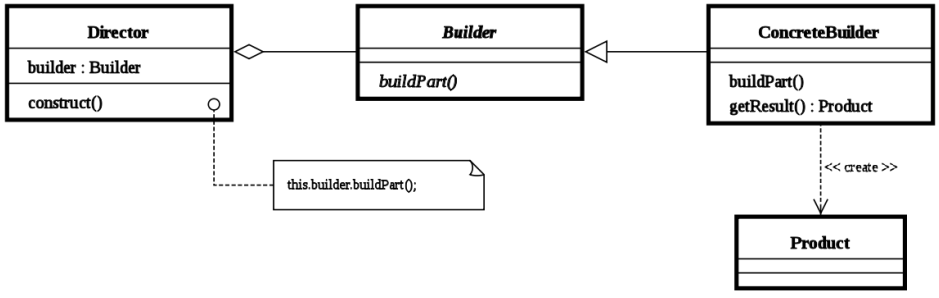
En el desarrollo existen unos patrones de diseño, que permite plantear una estructura adecuada, que hace al proyecto, sus servicio y microservicio a desempeñarse de forma óptima, y menos compleja sobre su funcionamiento, al seguir un patrón específico ayudar a seguir un interpretar y seguir un camino en el cual es más fácil tener un avance prolongado. Todo lo anterior siguiendo las respectivas normas. El objetivo de los patrones de diseño es facilitar el desarrollo al mismo tiempo que el trabajo en equipo, estandarizando la estructura del código y complementando con un estilo de codificación cuyo propósito es dar mayor interpretación. Entre los patrones de diseño tenemos,

Observador (**Observer**): Es un patrón de diseño de software que define una dependencia del tipo uno a muchos entre objetos, de manera que cuando uno de los objetos cambia su estado, notifica este cambio a todos los dependientes. Se trata de un patrón de comportamiento (existen de tres tipos: creación, estructurales y de comportamiento), por lo que está relacionado con algoritmos de funcionamiento y asignación de responsabilidades a clases y objetos.



Tomado de https://es.wikipedia.org/wiki/Observer_(patr%C3%B3n_de_dise%C3%B1o)

Constructor (builder): Se usa para permitir la creación de una variedad de objetos complejos desde un objeto fuente, el objeto fuente se compone de una variedad de partes que contribuyen individualmente a la creación de cada objeto complejo a través de un conjunto de llamadas secuenciales a una implementación específica que extienda la clase Abstract Builder.



Tomado de https://es.wikipedia.org/wiki/Builder_(patr%C3%B3n_de_dise%C3%B1o)