## INVESTIGACIÓN GITHUB, GRADLE Y MAVEN

**POR**

**ASTRID CAROLINA DÍAZ GÓMEZ**

**LUIS CARLOS MARIN CAMPOS**

**ANDRES DARIO HIGUITA PEREZ**

**LUIS MATEO HINCAPÍE MARTÍNEZ**

**ARQUITECTURA DE SOFTWARE**

**INTRUCTOR:**

**ROBINSON CORONADO GARCIA**

**INGENIERÍA DE SISTEMAS**



**UNIVERSIDAD DE ANTIOQUIA**

**2020-1**

**Introducción**

La importancia de los requisitos no funcionales, toman fuerza a la hora de llevar un producto de calidad a un usuario, logrando su mantenibilidad y escalabilidad.

## Objetivo General

Nuestro objetivo general es conocer los conceptos de los requisitos no funcionales, los cuales serán un factor importante en el momento de hacer parte de un equipo de desarrollo de alto rendimiento.

## Objetivos Específicos

Como objetivos específicos nos dispondremos a investigar sobre los siguientes requisitos no funcionales:

* Testability
* Reusability
* Reliability
* Integrability
* Legibilidad
* Fault Tolerance
* Open Source

# Procedimiento

Integrability

Características:

* Integralidad significa la capacidad de hacer que los componentes desarrollados por separado de un sistema funcionen juntos correctamente.
* La integralidad está relacionada con la interoperabilidad y nuevamente, la interconectividad.
* La integralidad tiene un impacto decisivo en el desarrollo y evolución de un sistema, por lo que debe tenerse en cuenta, así como las demás características de una familia de sistemas, como los requisitos de dominio, los elementos arquitectónicos de grano grueso y las prácticas utilizadas para el desarrollo y desarrollo. mantener una familia de sistemas y obtener productos de ella.

La integrabilidad depende de:

* complejidad de las componentes
* mecanismos y protocolos de comunicación
* claridad en la asignación de responsabilidades
* calidad y completitud de la especificación de las interfaces

Testability:

* Facilidad con la cual el software puede mostrar sus defectos (típicamente a través de pruebas de ejecución).
* Probabilidad de que, suponiendo que el software tiene al menos un defecto, fallará en la siguiente prueba.
* Condiciones necesarias:
  + controlabilidad - poder controlar el estado interno de las componentes
  + observabilidad - poder observar las salidas (outputs)

Reusability:

* Es la capacidad que tiene un sistema para que su estructura o alguna de sus componentes puedan ser usadas en futuras aplicaciones.
* La reusabilidad se relaciona con la arquitectura en que las componentes son las principales unidades de reutilización.
* la reusabilidad dependerá del acoplamiento de la componente, reusar una componente implica reusar la clausura transitiva de todas las componentes dependientes en la estructura de uso.
* La reusabilidad es una forma particular de modificabilidad

Ejemplo:

Un sistema S está compuesto de 100 componentes (S1 a S100). Se construye otro sistema T, y se descubre que las n primeras componentes son idénticas y pueden reutilizarse.

Legibilidad

# La legibilidad del código es conseguir un software que esté a la altura de codificación que permitan tener una [deuda técnica](https://jummp.wordpress.com/2010/05/12/la-deuda-tecnica-en-el-desarrollo-de-software/) aceptable y faciliten la evolución del sistema de información.

# Escribiendo el código usando patrones de diseño y con el nombramiento de las clases y variables.

Un programa puede hacerse más legible dándole cierto formato al código, utilizando el sangrado (indentación) para reflejar las estructuras de control del programa, e insertando espacios o tabuladores.

Fault Tolerance

Open Source

**Webgrafía**

https://jummp.wordpress.com/2011/08/27/desarrollo-de-software-la-legibilidad-del-codigo/