



# **REQUISITOS NO FUNCIONALES. ARQUITECTURA DE SOFTWARE**

## **INTEGRANTES.**

**ALEJANDRO BUILES MURILLO**

**MARIO DE JESUS PUENTES SANCHEZ**

**CARLOS MARIO FONSECA**



# Efficiency

## ¿Qué es la eficiencia del software?

- Muchas plataformas de software dicen que tienen beneficios de eficiencia, pero ¿qué significa eso exactamente? En muchos casos, se trata de tiempo y dinero, pero al seleccionar el software adecuado para implementar estos procesos, la eficiencia tiene que estar mucho más definida.





Para lograr un desarrollo de software eficiente, hay que seguir una serie de pasos:

- Definir las ineficiencias: En primer lugar, debe identificar las ineficiencias actuales dentro del software y los procesos comerciales actuales.
- Desarrollar o mejorar la documentación: No sabrá qué hacer si no tiene instrucciones sobre cómo hacerlo.
- Colaborar: Aunque los equipos cruzados y los procesos colaborativos suenan bien, las empresas aún tienen problemas con la implementación.
- Transparencia: Las partes interesadas provienen de todas las áreas de la organización, no solo de la alta dirección. Al desarrollar nuevos sistemas y software, es fundamental obtener información de quienes utilizarán el software.

# Pruebas de eficiencia en software

La eficacia de las pruebas de software cubre tres aspectos:

1. En qué medida el sistema satisface los requisitos del cliente.
2. Qué tan bien se cumplen las especificaciones del cliente por el sistema.
3. Cuánto esfuerzo se pone en desarrollar el sistema.





Algunas fórmulas para calcular la eficacia de la prueba de software (para diferentes factores):

- Eficiencia de la prueba =  $(\text{número total de defectos encontrados en la unidad} + \text{integración} + \text{sistema}) / (\text{número total de defectos encontrados en la unidad} + \text{integración} + \text{sistema} + \text{prueba de aceptación del usuario})$
- Prueba de eficiencia =  $(\text{No de defectos resueltos} / \text{No total de defectos presentados}) * 100$



# Mobility

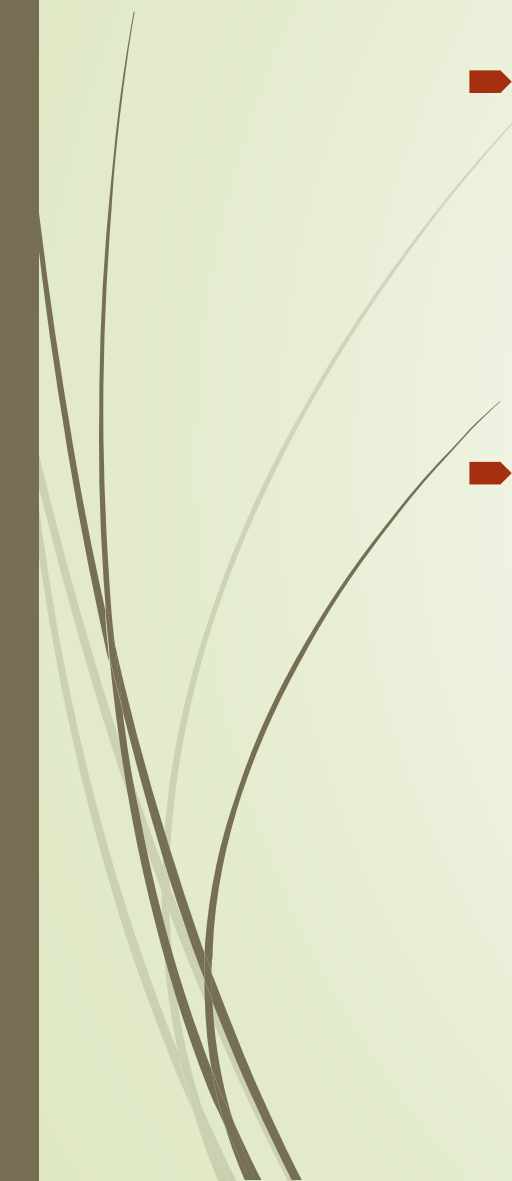
## ¿Qué es Mobility y por qué es importante?



- Mobility significa proveer de acceso universal y “multi-modal” a las herramientas y aplicaciones en las que confía para ser productivo con independencia de dónde se encuentre o a qué dispositivo tenga acceso en ese momento. El acceso a todos los mensajes -voz, e-mail, fax, e-fax- desde un mismo buzón; marque un sólo número para localizar compañeros, esté en la oficina o fuera de ella; y disfrute de las mismas prestaciones desde donde esté trabajando (en un viaje, en casa, en la oficina).



## ¿Por qué mobility es tan importante para las empresas?

- Las comunicaciones empresariales están en continuo crecimiento, y las posibilidades de comunicación han aumentado vertiginosamente. La movilidad ayuda a ofrecer unas comunicaciones universales, facilitando que las personas se comuniquen de la forma que ellos elijan.
  - Gestionar las comunicaciones de forma efectiva e integrar las comunicaciones con los procesos diarios es clave para mejorar la productividad y los procesos del negocio. Por ejemplo, si una persona de servicios reporta la solución de un problema en la base de datos, el sistema llama al cliente, le informa de la situación y le informa del coste asociado. El cliente recibe la información más rápido y la compañía recibe el pago más rápidamente.
- 

# Customizability



- La personalización es "un medio que busca satisfacer las necesidades de los clientes de manera más efectiva y eficiente, haciendo que las interacciones sean más rápidas y fáciles para que, en consecuencia, aumente la satisfacción del cliente y la probabilidad de visitas repetidas", según TechTarget.
- Se logra cuando un sistema adapta una experiencia basada en el comportamiento previo de un consumidor. Por ejemplo, Amazon personaliza su página de inicio para cada usuario en función de las búsquedas, vistas y compras anteriores de ese usuario. La customización, por definición, es "la acción (por parte de un usuario) de modificar algo para adaptarlo a un individuo o tarea en particular".





# Degradation of service

- Desde la capa de infraestructura hasta el front-end de la aplicación, su sistema tiene límites. Y cuando lo empuja más allá de sus umbrales, su sistema se comporta de manera inesperada. Los servicios se degradan. Ocurren cortes. Y aunque a menudo pensamos en la respuesta a incidentes en términos de encontrar y corregir la causa raíz del problema, en muchos casos, no existe una sola causa. Un aumento repentino en la demanda de un servicio puede provocar la acumulación de una gran cantidad de solicitudes en una cola de mensajes, lo que normalmente no es un problema. Pero si combina eso con un parámetro mal configurado que evita que el clúster de cola de mensajes aumente los recursos, se queda con un servicio degradado y posiblemente interrumpido.

# code-space-perfomance



- ▶ En general, un programa de computadora puede optimizarse para que se ejecute más rápidamente, o para hacerlo capaz de operar con menos almacenamiento de memoria u otros recursos, o para consumir menos energía.
- ▶ Dado un diseño general, viene a continuación una buena selección de algoritmos y estructuras de datos eficientes, y la implementación eficiente de estos algoritmos y estructuras de datos..

- 
- 
- Después del diseño, la elección de algoritmos y estructuras de datos afecta la eficiencia más que cualquier otro aspecto del programa. Generalmente, las estructuras de datos son más difíciles de cambiar que los algoritmos, ya que un supuesto de estructura de datos y sus supuestos de rendimiento se utilizan en todo el programa, aunque esto puede minimizarse mediante el uso de tipos de datos abstractos en las definiciones de funciones y manteniendo restringidas las definiciones de estructura de datos concretas. a algunos lugares.
  - Mientras desarrollamos un programa o aplicación ponemos en práctica nuestro proceso de trabajo. Mientras que ciertas fases del proceso pueden ser eficientes en proyectos pequeños, cuando nos enfrentamos con proyectos a gran escala nos damos cuenta que los procedimientos no son tan efectivos como deseábamos y que cambiando ciertos procesos podríamos avanzar con mayor rapidez. Como resultado, generamos prácticas eficientes para gestionar nuestro código en este tipo de proyectos y empleamos este conocimiento para cambiar nuestro proceso de trabajo y hacerlo más efectivo.

# Hardware Cost

- Debido al hecho de que la exploración del espacio de diseño es un proceso iterativo y se analizan una gran cantidad de arquitecturas diferentes, la estimación de costos debe ser rápida.
- Además, el método de estimación debe ser independiente de la tecnología, es decir, las mismas herramientas deben funcionar cuando se cambia la tecnología objetivo. Finalmente, la exploración se utiliza para seleccionar candidatos para optimizaciones adicionales, por lo que se necesita el costo relativo en lugar del costo absoluto.





# Navegability

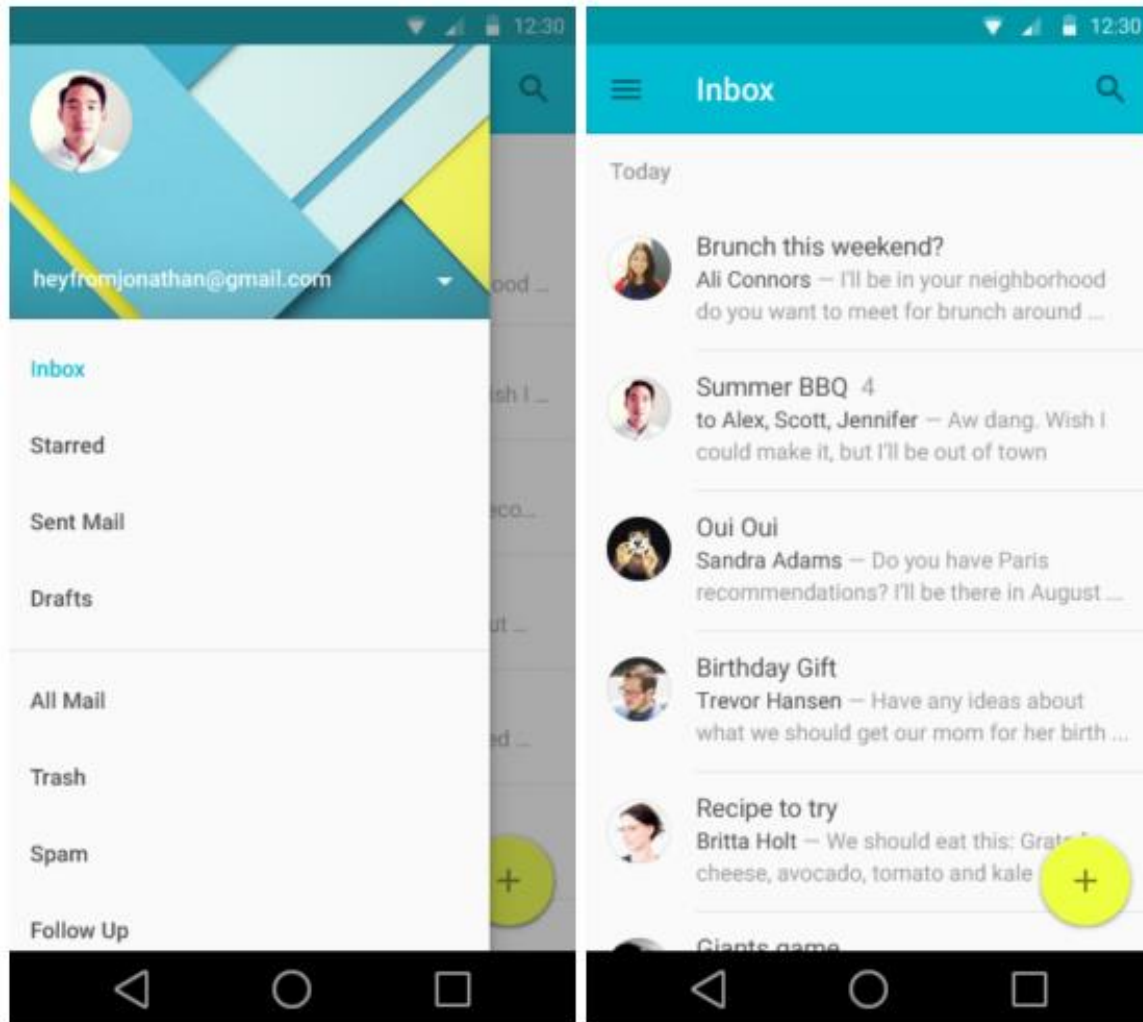


- La navegabilidad es la facilidad con la que un usuario puede desplazarse por todas las partes que componen un software. Para lograr este objetivo, un software debe proporcionar un conjunto de recursos y estrategias de navegación diseñados para conseguir un resultado óptimo en la localización de la información y en la orientación para el usuario.
- Las interfaces de navegación tienen que ayudar a los usuarios a responder a tres preguntas fundamentales relacionadas con la navegación:
  1. ¿Dónde estoy?
  2. ¿Dónde he estado?
  3. ¿Dónde puedo ir?



# Patrones básicos de Navegación en Apps Móviles

- La Navegación en tu app debe ser intuitiva y predecible. Tanto para usuarios nuevos como para quienes vuelven a usarla debe ser simple de descubrir como moverse en las distintas secciones con comodidad. Pero lograr que la navegación sea descubrible y accesible puede ser todo un desafío por las limitaciones de las pantallas pequeñas y la necesidad de priorizar el contenido sobre los componentes de la interface. Diferentes patrones de navegación apuntan a resolver este mismo problema de diferentes modos, pero todos tienen, en algún aspecto, problemas de usabilidad.



## Hamburger Menu (menú de hamburguesa)

- El espacio en la pantalla es “mercancía valiosa” en un móvil y el hamburger menu (o menu lateral) es uno de los patrones más comunes para ahorrar espacio. Este menu nos permite ocultar la navegación en el borde izquierdo de la pantalla y se muestra solo luego de una acción del usuario. Puede ser particularmente útil si queremos que el usuario se concentre en el contenido principal.

## Pros

- *Espacio para una gran cantidad de elementos a mostrar.* La principal ventaja de este tipo de menu es que puede contener cómodamente una larga lista de opciones de navegación en un espacio pequeño.
- *Diseño limpio.* Libera espacio en la pantalla al colocar todos los elementos relacionados a la navegación en un menu desplegable.

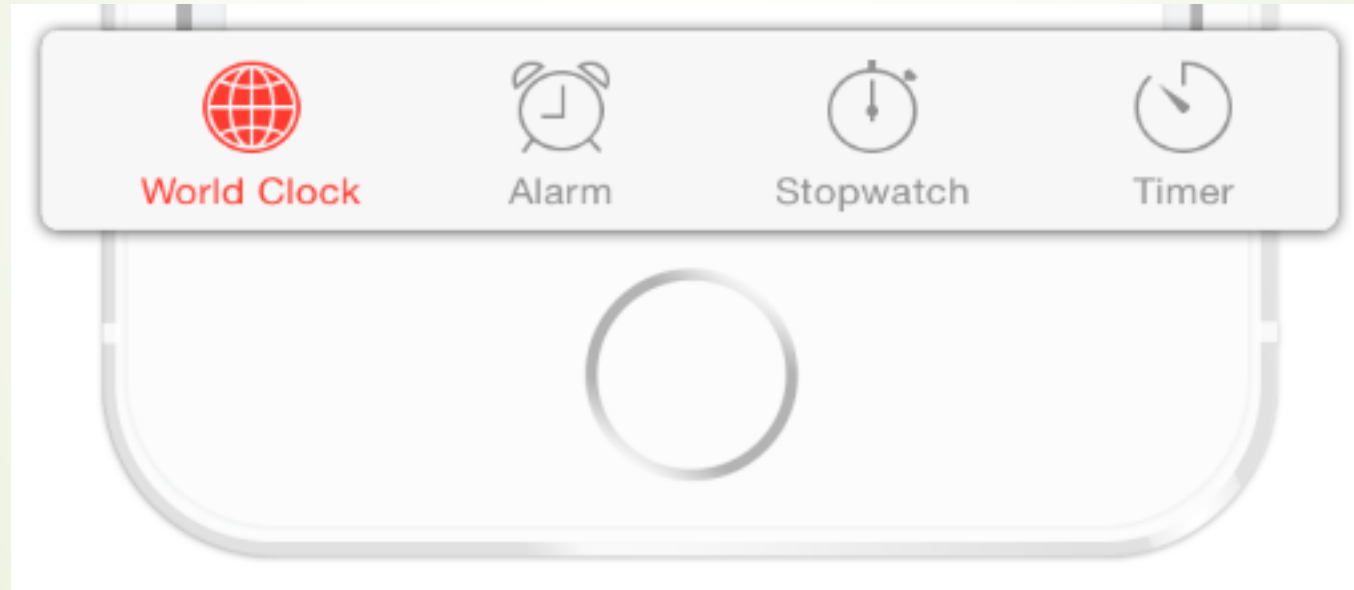
## ➤ Cons

- *Es menos “descubrible”.* Lo que está fuera de la vista, esta fuera de la mente. Cuando la navegación está oculta, los usuarios son menos propensos a utilizar la navegación. Aún cuando esta opción parece convertirse en un estándar, muchos usuarios sencillamente ni siquiera piensan en abrirlo.
- *Colisiona con reglas de navegación de varias plataformas.* El hamburger menu se convirtió en un estándar en Android (con el nombre de Navigation drawer en Material Design), pero en iOS no puede implementarse sin colisionar los elementos básicos de navegación, y esto puede sobrecargar la barra de navegación



# Tab Bar

- Este patrón fue heredado del diseño de interacción para *Desktop*. Usualmente contiene pocas opciones de importancia similar que requieren acceso directo desde cualquier parte de la app.



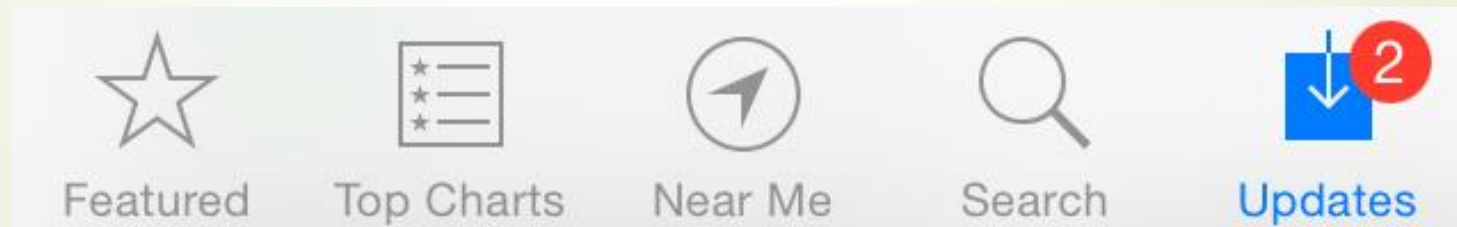


## Ejemplo

- El Tab bar de Twitter permite a los usuarios navegar directamente a las pantallas asociadas con los iconos.

## Pros

- *Comunica de manera rápida y sencilla la posición actual del usuario.* Con pistas visuales correctamente utilizadas (iconos, etiquetas y colores) no es necesario ninguna explicación extra.
- *Los Tab bars persisten.* Muestra todas las opciones de navegación en la pantalla todo el tiempo. El usuario tiene una clara visibilidad de todas las secciones y un acceso rápido a ellas.



## Cons

- Número limitado de opciones. Si tu app tiene más de cinco opciones sería muy difícil de acomodarlas en un navegación bar respetando el tamaño “óptimo de objetivo de toque”.

