CLASE 2

Necesidad vs Diferencias

Repaso

¿Qué es el software?

Todo comienza en la necesidad de un cliente / usuario y el software aparece como **producto / solución / herramienta** de esa necesidad de un cliente / usuario y el software aparece como **producto / solución / herramienta** de esa necesidad necesidad

¿Qué es el Testing?

Las **pruebas de software** (*testing*) son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar **información** objetiva e independiente sobre la **calidad del producto** a la parte interesada.

¿Qué son los requerimientos?

Es una definición de lo que debe y no debe hacer el software. Detallan cómo debe comportarse el software para satisfacer una necesidad de un usuario.

Tipos de requerimientos

Funcionales

Son los requerimientos que **definen las funciones** que el sistema será capaz de realizar → que el usuario pueda ingresar un pedido.

No Funcionales

Se trata de requisitos que no **se refieren** directamente a las funciones específicas del sistema, sino **a las propiedades del sistema**: rendimiento, seguridad, usabilidad. En palabras más sencillas, no hablan de “lo que” hace el sistema, sino de “cómo” lo hace.

Compartimos los resultados de la actividad de la clase anterior

- Pensar en distintos objetos / elementos y elijan uno (ejemplo una pelota, guitarra, etc)
- Identificar necesidad (en base a una necesidad se crea el objeto) / objetivo de ese elemento
- contar como verificarán que se cumpla esa necesidad (si la necesidad es entretener, podemos decir que el objeto pelota cumplió la necesidad para la que fue creada, que es entretener)

ESRE

ESPECIFICACIÓN FUNCIONAL DE REQUERIMIENTOS

Es un **documento que detalla todos los requerimientos** (requisitos) que el software debe cumplir, por lo que se constituye como una herramienta fundamental al momento de hacer el testing. Esta documentación debe ser:

● Completa ● Inequívoca ● Priorizable ● Modificable ● Trazable

ESRE

ESPECIFICACIÓN FUNCIONAL DE REQUERIMIENTOS

● **COMPLETA.** Todos los requerimientos deben estar reflejados en ella.

● **INEQUÍVOCA.** La redacción debe ser **clara** de modo que no se pueda malinterpretar.
Esto es digamos una utopía, porque cualquier redacción puede dar lugar a malinterpretaciones. Teniendo esto en cuenta es que se **intenta incluir diagramas, prototipos y** la mayor **información** posible para tratar de minimizar las malas interpretaciones.

● **PRIORIZABLE.** Los requerimientos deben poder **organizarse jerárquicamente** según su relevancia para el negocio y clasificándolos en esenciales, condicionales y opcionales.

● **MODIFICABLE.** Aunque todo requerimiento es modificable, se refiere a que debe ser fácilmente modificable.

● **TRAZABLE.** Se refiere a la posibilidad de **verificar la historia**, ubicación o aplicación de un ítem a través de su identificación almacenada y documentada.

ESRE

ESPECIFICACIÓN FUNCIONAL DE REQUERIMIENTOS

EJEMPLO

Buscador de productos de una aplicación de compras online:

“La pantalla de búsqueda tendrá un campo en la parte superior de la pantalla, donde el usuario podrá escribir el texto que está buscando – al lado habrá un botón que permitirá “buscar”. Cuando el usuario hace clic en el botón de buscar, se desplegará debajo el resultado de la búsqueda que incluye todos los productos que contengan la palabra buscada en el nombre del producto o la descripción. El resultado de la búsqueda estará ordenado alfabéticamente e incluirá 20 elementos por cada página. La grilla de resultados se podrá navegar en forma estándar – moverse entre las diferente páginas, incluir número de ítems en la lista, número de página en que el usuario se encuentra, etc.”

HISTORIAS DE USUARIO

Otra técnica para definir requerimientos. A diferencia del documento de requerimientos, las historias de usuario intentan capturar suficiente información para que sea posible que **desarrolladores y cliente tengan una conversación** para entender qué es lo que se intenta resolver.

La historia de usuario responde las siguientes preguntas

- ¿**A quién** le importa?
- ¿**Qué debe** hacer en concreto?
- ¿**Para qué**?

HISTORIAS DE USUARIO

Ejemplo

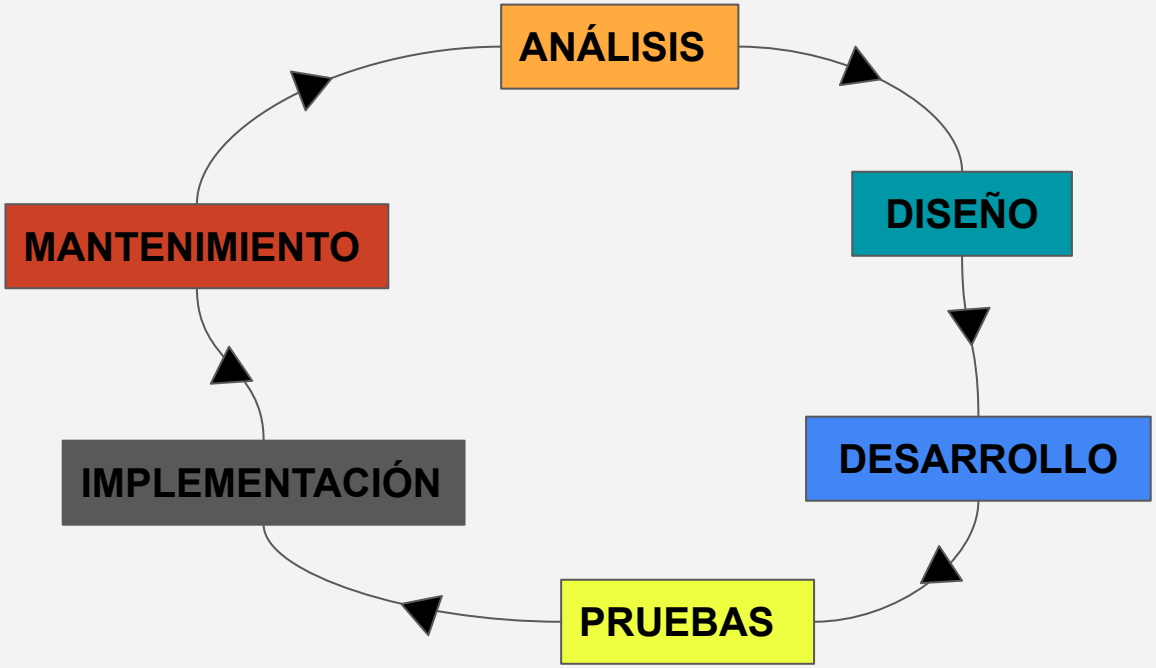
Para la misma funcionalidad una historia de usuario diría lo siguiente:

“Como un comprador quiero poder buscar un producto por un texto contenido en su nombre o descripción de forma de no tener que recordar cómo se llama exactamente”

ESRE vs HISTORIAS DE USUARIO

Ambas detallan que el usuario que quiere comprar un producto en el sitio de carrito va a poder buscar productos por un texto, pero el nivel de detalles del resultado esperado es diferente en el ESRE (donde la idea es que este bien claro que y como debe funcionar el software) de la historia de usuario donde solo se explica el qué y por qué, por lo que se deja que el equipo de proyecto defina el cómo junto con el cliente.

CICLO DE VIDA DEL SOFTWARE



ANÁLISIS

Hay que averiguar qué es exactamente lo que tiene que hacer el software. La etapa de análisis corresponde al proceso a través del cual se intenta descubrir **qué es lo que realmente se necesita** y se llega a una comprensión adecuada de los requerimientos del sistema (las características que el sistema debe poseer).

DISEÑO

Se estudian las posibles implementaciones que hay que construir y la estructura general del software.
Es una etapa complicada, y si la solución inicial no es la más adecuada, habrá que redefinirla.

DESARROLLO

Producción del sistema software

PRUEBAS

Como errar es humano, la fase de pruebas del ciclo de vida del software busca **detectar los fallos cometidos** en las etapas anteriores para corregirlos. Por supuesto, lo ideal es hacerlo antes de que el usuario final se los encuentre. Se dice que una prueba es un éxito si se detecta algún error.

IMPLEMENTACIÓN

Se trata de **elegir las herramientas** adecuadas, un **entorno** de desarrollo que haga más sencillo el trabajo **y el lenguaje** de programación óptimo.
Esta decisión va a depender del diseño y el entorno elegido.

MANTENIMIENTO

Esta es una de las fases más importantes del ciclo de vida de desarrollo del software. Puesto que el software ni se rompe ni se desgasta con el uso, su mantenimiento incluye tres puntos diferenciados:

- Mantenimiento **correctivo**
  Eliminar los defectos detectados durante su vida útil.

- Mantenimiento **adaptativo**
  Adaptarlo a nuevas necesidades.

- Mantenimiento **perfectivo**
  Añadirle nuevas funcionalidades.

CELFAR

Celfar es una app que convierte de grados Celsius y Fahrenheit:

Veamos las especificaciones y luego identifiquemos las necesidad de la Aplicación

https://nahual.github.io/qc-celfar/especificaciones.html

Luego de ver las especificaciones e identificar las necesidades de la aplicación, vamos a ver la primera versión

Versión 1:

https://nahual.github.io/qc-celfar/?v=1

- Identifiquemos las diferencias entre las especificaciones y el desarrollo que vimos.
- Identificamos los errores

Hablemos de Versiones…

El versionado de software es el proceso de asignación de un nombre, código o número único, a un software para indicar su nivel de desarrollo.

Volvamos A CELFAR…

Ya vimos una primera versión y pudimos identificar algunos errores. Pensemos que esos errores fueron reportados al equipo de desarrollo, y ellos realizaron las modificaciones necesarias para corregirlos.
Luego nos informan que hay una segunda versión del desarrollo, la cual está disponible para probar. Tendríamos que verificar que los errores reportados en la primera versión fueron corregidos, y deberíamos revisar que no haya errores nuevos

Mostramos la versión 2 de la app para mostrar mejoras entre la versión 1 y 2

https://nahual.github.io/qc-celfar/?v=2