CURSO

# SOFTWARE TESTING

UNIDAD 10 – Automatización

UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# PRESENTACIÓN

En esta unidad repasaremos nociones básicas de automatización



UTN.BA
UNIVERSIDAD TECNOLÓGICA NACIONAL
FACULTAD REGIONAL BUENOS AIRES

# TEMARIO

1. – Definición
2. – Como utilizarlo
3. – Cuando utilizarlo.
4. – Ejemplos.

# ¿Testing Automatizado o Manual ?

- La automatización es una *herramienta* más dentro del testing cuyo principal objetivo es la reducción de tiempos → En ningún caso sustituye otras disciplinas del testing.
- Lo que automatizamos son chequeos, comprobaciones que los testers manuales previamente han detectado antes: ciertas pruebas de regresión, smoke test, etc.
- La automatización:
  - Elimina el error humano
  - Permite la re-ejecución de la prueba en las mismas condiciones
  - Pruebas menos complejas que las manuales que garantizan el funcionamiento básico de procesos de negocio de extremo a extremo
  - Reutilización: Misma prueba con distintos lotes de datos, sobre distintos navegadores, componentes reutilizables, etc.

# Automatización

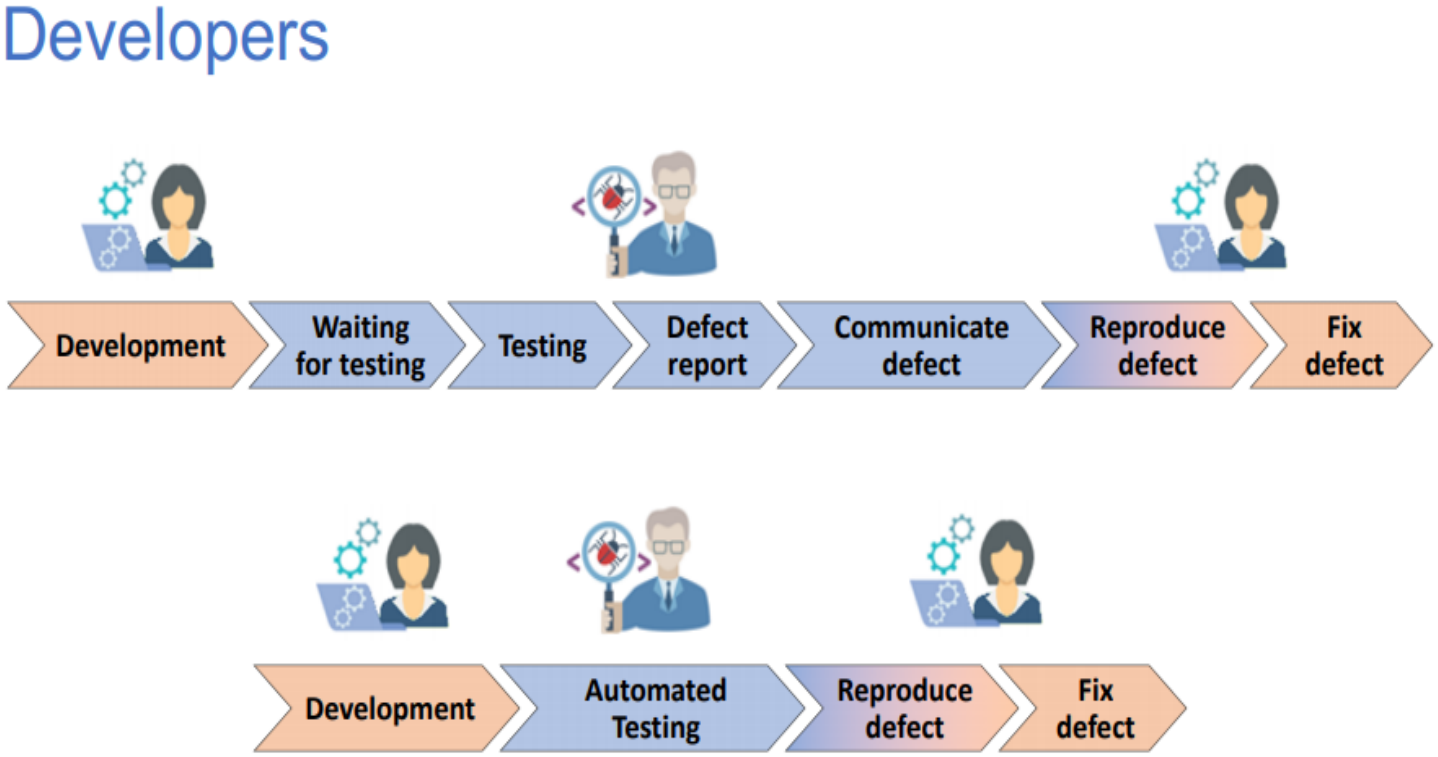
**Mito: La automatización es una tarea sencilla**

Hay que tratarla como un proyecto de SW dotándola de:

- Tiempo y recursos.
- Personal especializado.
- Planificación, diseño, desarrollo, prueba.
- Es fundamental realizar la gestión, mantenimiento y evolución del test.
- Si se realizan cambios en el sistema que puedan afectar la ejecución de las pruebas, éstas también se tienen que actualizar.
- En general, en el ambiente del software se conocen los *beneficios* de disponer de pruebas automatizadas. Sin embargo, es menos común entender los costos que esto implica. No están claras las características de la *inversión* asociada.

# Productividad

La automatización requiere un conjunto de actividades iniciales que hacen que sus resultados sean visibles a medio y largo plazo
La ejecución de pruebas manuales es mucho más dinámica y rápida en tiempo de respuesta
La productividad de la automatización está en la re-ejecución
Convivencia, debe haber un equilibrio entre las pruebas manuales y automáticas



# Ventajas

- La automatización de pruebas es una de las inversiones que produce los mejores y más satisfactorios resultados (ROI)
- Proporciona información “real / objetiva ” de los criterios de calidad:
  - “Sí, el producto está preparado para salir”
  - “No, no está preparado y esta es la razón por la que no puede salir”
- Reducción del costo de las pruebas en el medio-largo plazo
- Incremento de la cobertura de las pruebas
- Reducción del tiempo de ejecución de los ciclos de pruebas
- Permite un desarrollo iterativo
- Disminución de los errores humanos

# Desventajas

- No es posible automatizar el 100% de las pruebas, o al menos no es rentable.
- Es complicado volver al estado original en que se realizaron las pruebas
- Generar un buen lote de datos de prueba
- Contar con una buena herramienta de automatización
- Se requiere tiempo y esfuerzo tal como si fuera un proyecto de desarrollo de SW:
  - Planificación y Diseño: Conocer qué queremos probar (entradas) y definición del plan de pruebas
  - Implementación: Planificar qué merece la pena automatizar
  - Ejecución: definir el set de datos a utilizar y ejecutar desatendidamente un conjunto de pruebas
  - Evaluación : ¿A qué se debe el fallo en una prueba?  Distinguir entre:
    - Fallo en el diseño e implementación de la prueba
    - Fallo de la aplicación a probar

# ¿Que pruebas automatizamos?

- Probablemente lo que primero suele venirnos a la cabeza cuando oímos hablar de automatización de pruebas son automatizaciones de “record & play”
- Depende de qué plataforma, a primera vista pueden resultar las más sencillas de automatizar.
- Es cierto que son automatizaciones necesarias, pero no son las que más retorno de inversión aportan, ni las que deberían automatizarse en mayor cantidad.
- Principalmente, porque la interfaz de usuario es la parte más propensa a cambios de toda la aplicación.

# ¿Que pruebas automatizamos?

- Para automatizar pruebas y tener fiabilidad sobre lo que estamos ejecutando necesitamos cierta estabilidad: un cambio en la interfaz podría hacer fallar la prueba automática, y en ese caso, tendríamos que readaptarla para que volviera a funcionar.
- Por eso la automatización de pruebas de Interfaz de Usuario suelen tener un coste de mantenimiento mayor que el resto
- Por ejemplo, un buen primer paso es automatizar los test de API (funciones, elementos que ofrecemos desde nuestro software para que otro software, u otras partes del nuestro, puedan interactuar con él).
- Hay varios niveles a la hora de automatizar pruebas. El secreto está en automatizar en el grado adecuado y en los niveles adecuados para nuestra aplicación.

# Piramide de Cohn

En ella Cohn establece que hay varios niveles de pruebas, y señala el grado en el que deberíamos automatizarlas. Lo ideal sería:

**Muchos tests unitarios automáticos:**

Porque un primer punto primordial para detectar fallos es a nivel de desarrollador. Si una funcionalidad en este punto falla, podrían fallar pruebas de los siguientes niveles: integración, API etc.

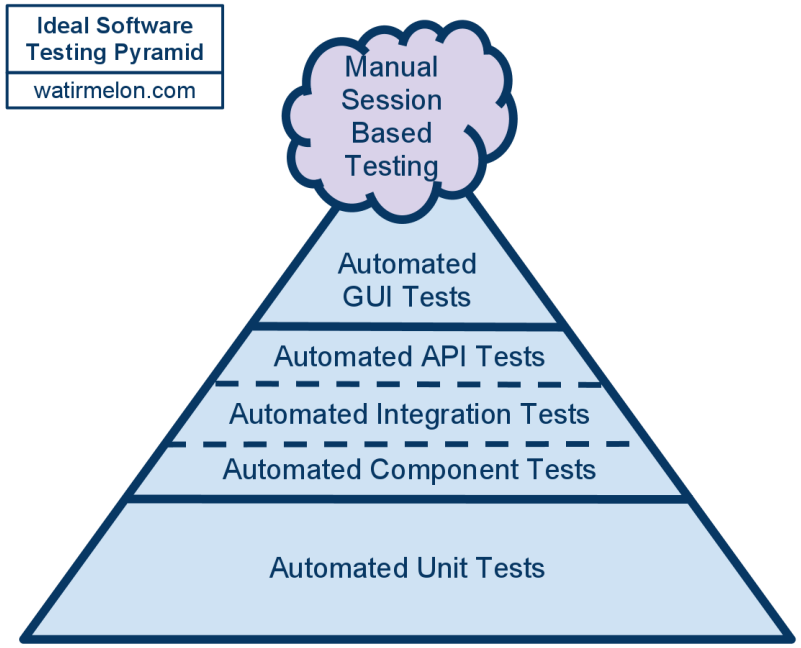
Además es mucho más sencillo detectar el origen de un error en esta etapa

**Bastantes tests a nivel de API, integración de componentes, servicios**:

Son los más estables y candidatos a automatizar. Si cambia la Interfaz gráfica, el servicio / componente de integración debería mantenerse sin cambios.

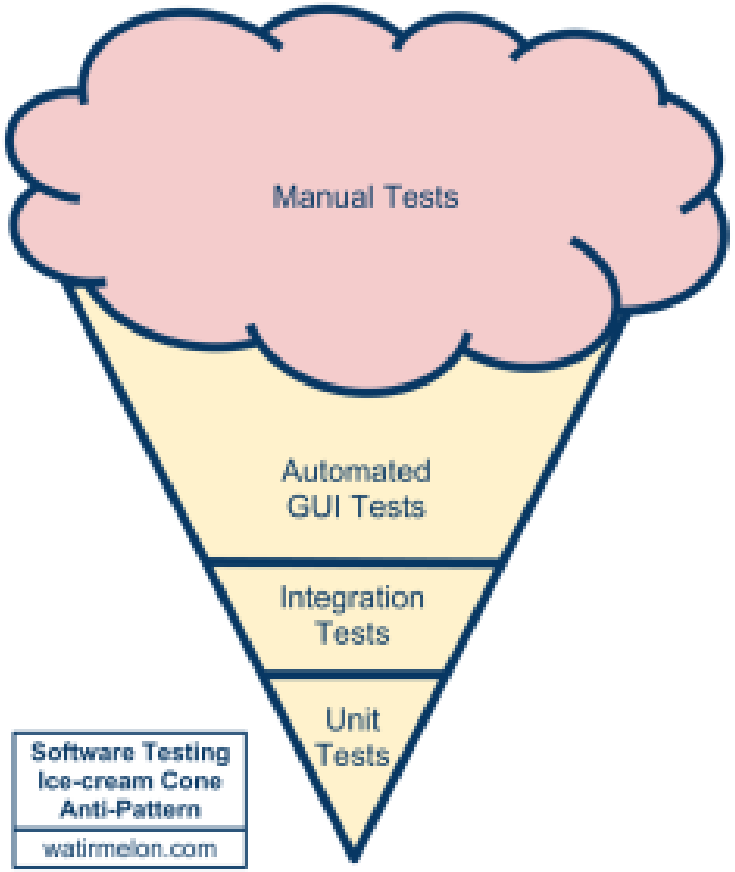
**Menos tests de interfaz gráfica automatizados:**

Ya que estos tests son variables, lentos en su ejecución, difíciles de mantener actualizados y con muchas dependencias con otros componentes.



# Piramide de Cohn – Patrón cono de helado

- Es lo contrario a la pirámide de Cohn: centramos el foco en muchas pruebas manuales y en automatizar pruebas de interfaz de usuario, y nada de pruebas unitarias. Con todos los problemas que acarrea no detectar errores en los otros niveles y dejarlo todo para la parte más visible de la aplicación.
- Además hay que tener en cuenta, que una buena estrategia de automatización de pruebas conlleva más cosas que solo automatizar las pruebas en sí:
  - crear un buen framework de automatización, parametrizar los tests, las ejecuciones para los distintos entornos,
  - lanzar los test con distintos datos de prueba y gestionar dichos datos, sistemas de logs,
  - reportes con información que sirvan para obtener conclusiones,
  - montar una buena infraestructura contra la que lanzar esos tests, paralelizarlos etc.



# Etapas del proceso

## ¿Cómo empezamos?

- Podemos distinguir 2 etapas en el proceso:
  1. Desarrollo de la plataforma de automatización
  2. Desarrollo de las pruebas automatizadas

## 1. Desarrollo de la plataforma de automatización

Este es el puntapié inicial en la automatización de pruebas para un producto puntual.

- En primera instancia es vital conocer los requerimientos necesarios para las pruebas. Por ejemplo, verificaciones con lectura de bases de datos o si hay reportes en pdf que las pruebas deben leer y comparar.

# Etapas del proceso

## 1. Desarrollo de la plataforma de automatización (Cont.)

- Luego que sabemos lo que necesitamos buscamos herramientas o la combinación de estas para satisfacer las necesidades. En el ejemplo anterior (lectura en bases de datos y pdf’s) podemos combinar un framework de pruebas unitarias como JUnit o TestNG con librerías de automatización a nivel de interfaz gráfica, librerías para conexión a bases de datos, librerías para manejo de pdf, entre otras.
- Una vez seleccionadas las herramientas es necesario integrarlas y hacer una prueba de concepto de la plataforma.
- El siguiente paso es desarrollar las funciones o módulos básicos para que las pruebas sean composición de estas. Por ejemplo, crear las funciones: Login, CrearInstancia, VerificarCreación, ModificarInstancia, VerificarModificación, Logout.

# Etapas del proceso

## 1. Desarrollo de la plataforma de automatización (Cont.)

- Las pruebas serán combinaciones de estas funciones con diferentes datos. Por ejemplo:
  - Login (usuario1)
  - CrearInstancia (instancia1)
  - VerificaciónCreación (instancia1)
  - Logout
  - Login (usuario2)
  - ModificarInstancia (instancia1)
  - VerificarModificación (instancia1)
  - Logout
- Por supuesto, en el desarrollo de las pruebas se continuará creando nuevos módulos y funciones a medida que surjan las necesidades.

# Etapas del proceso

## 2. Desarrollo de las pruebas automatizadas

- Como en un proyecto de desarrollo de software (que de hecho lo es), comenzamos con ciclos de diseño, implementación, verificación y validación de pruebas. Es importante marcar ciclos cortos de manera que se puedan mostrar resultados rápidamente.
- Un esfuerzo no despreciable es el dedicado a la configuración del entorno de datos necesario para los scripts.
- El diseño de las pruebas es fundamental para encontrar elementos comunes y reutilizables que se puedan consolidar en funciones para el uso de varios scripts. De esta manera aumentamos la productividad porque se crearán nuevos scripts más rápido, reutilizando funciones. De la misma forma se aumenta la mantenibilidad ya que ante cambios en las interfaces de la aplicación los cambios se realizan únicamente en las funciones y no en cada script.

# Etapas del proceso

## 2. Desarrollo de las pruebas automatizadas (Cont.)

- Para la implementación, en este esquema de trabajo, las herramientas Record and Playback pasan a un segundo plano ya que directamente se codifica las pruebas pudiendo utilizar  scripts similares como plantillas. La grabación de un script no tiene sentido ya que este se construye a partir de la composición de funciones y no una nueva serie de acciones.
- En cuanto a la etapa de prueba de los scripts, cabe destacar que la  verificación se refiere a la prueba de los scripts en el ambiente en donde se desarrollan. La validación es deseable que se haga en un nuevo ambiente para probar así que los datos requeridos para la ejecución están bien documentados y configurados.

# PREGUNTAS

# Ejercicio en clase

Practica: Automatizar un test con Selenium WebDriver


Fin de a Unidad 10.