Capítulo 3: ¿ Qué debemos automatizar?

 ¡Automatiza todo!

Independientemente de si estás comenzando a automatizar un proyecto nuevo o uno existente, nunca debes considerar automatizar todo, solo por el simple hecho de automatizar. Es tentador, lo sé. Pero veamos por qué eso no es una buena idea:

* **La creación de *scripts* de automatización lleva tiempo; el tiempo es dinero:** si tu enfoque es automatizar todo, no estás obteniendo un buen retorno de su inversión (ROI).
* **Se requiere mantenimiento:** no importa qué tan bien esté escrita tu automatización; siempre requerirá mantenimiento. Además del mantenimiento del script, tendrás mantenimiento de cosas como herramientas, ejecuciones de pruebas, librerías de desarrollo, etc. Si automatizas todo, seguramente te verás bajo una avalancha de mantenimientos y no podrás asumir nuevas funciones.
* ***Scripts* defectuosos**: vinculados al ítem anterior de mantenimiento. Si tu automatización falla debido a la falta de mantenimiento, las personas, con el tiempo, perderán confianza en sus resultados.
* **Aumento de los tiempos de ejecución:** cada *script* deberá ejecutarse para agregar valor. Si tienes un gran conjunto de scripts por ejecutar, sus tiempos de ejecución aumentarán naturalmente, incluso si los ejecuta en paralelo aún verás un aumento en el tiempo de ejecución a medida que agregas más y más *scripts*.
* **Validación de fallas:** si ejecutas un montón de pruebas innecesarias cada vez, aumentará la necesidad de validar las fallas que pueda tener.

Vale la pena mencionar algunas cosas parecen atractivas de automatizar, desde el punto de vista del escenario, pero que no siempre funcionan bien cuando se implementan o no se pueden automatizar:

* Colores
* Captcha
* Accesibilidad
* Documentación
* Flujos realmente largos: considera profundizar en la parte que necesitas probar o hazlo en el *backend* o las APIs
* Seguir todos los flujos de usuario: considera automatizar el flujo de extremo a extremo o de punta a punta (E2E: *End-to-end*) una vez y luego divide en escenarios para probar piezas más pequeñas en la capa de integración.

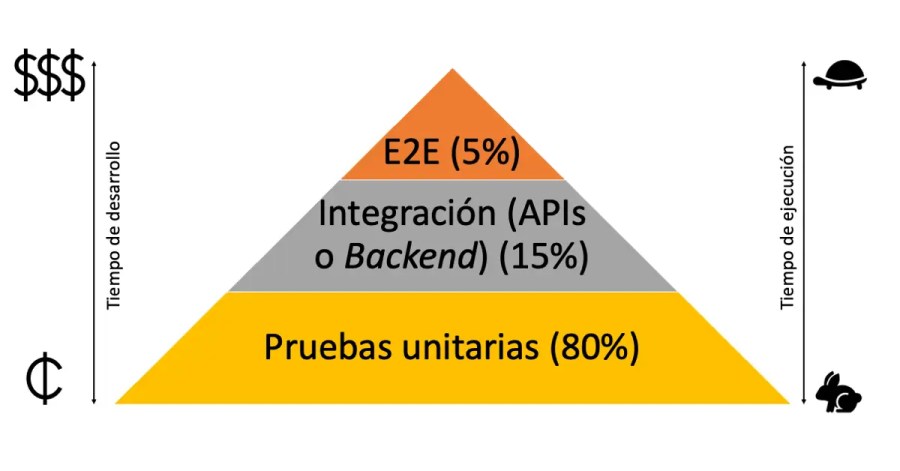
 ¡Automatiza lo que mas aporta valor!

Hemos hablado de por qué no deberías automatizar todo, ahora hablemos sobre qué *deberías*automatizar. Aquí hay algunas preguntas que puedes hacer para determinar que es un candidato ideal para la automatización:

* **¿Es este un escenario que los clientes experimentan a menudo?**Piensa en los principales flujos de usuarios de tus clientes. Por ejemplo, asegurarte de que los clientes puedan registrar una cuenta y comprar cosas sería importante si tienes una tienda en línea.
* **Si este escenario resulta en un defecto, ¿retrasaría un lanzamiento?**¿Con qué rapidez se solucionaría un defecto del escenario? ¿Le impediría desplegar el proyecto o, peor aún, inutilizaría tus entornos de prueba o producción?
* **¿El resultado de este escenario solo se obtiene por este camino?**Quizás haya más de una forma en que un cliente puede seguir el flujo. Si este está roto, ¿aún podrían realizar las tareas importantes?
* **¿Cuál es la complejidad de escribir este escenario?**¿Qué tan rápido puedes escribirlo? ¿Existen nuevas herramientas o tecnologías que serían necesarias para completarlo?
* **¿Qué tan estable es el área en la que reside este escenario?**¿Constantemente aparecen defectos en esta área?
* **¿Es esta una prueba que se ejecutará de manera constante?**Puede que no tenga sentido dedicar tiempo a automatizar algo que solo se ejecuta una vez al año.

Pruebas unitarias vs. Pruebas de integración vs. Pruebas de UI

Una parte del análisis de qué y cómo automatizar es determinar en qué capa automatizar el escenario de pruebas. Muchos gurús de las pruebas han publicado su propia versión de la pirámide de automatización, pero aquí está la pirámide original como referencia:

Pirámide de pruebas de Mike Cohn

Esencialmente, la pirámide visualiza el conjunto de automatización total y define el porcentaje de pruebas o esfuerzo de aseguramiento por cada capa. Este porcentaje es basado en promedios y cada organización lo ajusta a sus necesidades particulares, pero siempre se debe tener en cuenta que la mejor prueba es aquella que entrega retroalimentación lo más antes posible.

Pruebas unitarias

Las pruebas unitarias constituyen la capa más grande de la pirámide. Es una prueba simple que se centra en una única unidad (un fragmento de código aislado, por ejemplo, una clase). Esta prueba no debe depender de ningún sistema externo, como una base de datos, configuraciones o incluso otras pruebas.

|  |  |
| --- | --- |
| **Pros** | **Contras** |
| * Rápidas de escribir * Brinda a los desarrolladores comentarios inmediatos a medida que se ejecutan rápidamente * Brinda detalles específicos al desarrollador sobre el problema, ya que están contenidos en unidades individuales | * Dado que son específicos de una unidad, se pasan por alto errores en la integración y las capas de la interfaz de usuario |

Pros y contras de las pruebas unitarias

Para estas pruebas es muy común utilizar dobles de pruebas, mocks, stubs.

Prueba de integración

La capa de pruebas de integración está destinada a aumentar el alcance de las pruebas e incluye pruebas de nivel de servicio o API. Estas pruebas se basan en la capa de pruebas unitarias, garantizando que las unidades de código se puedan integrar sin problemas. Esta capa también debe considerar cualquier prueba de contrato que pueda tener. Por ejemplo, cualquier integración de terceros con su aplicación.

|  |  |
| --- | --- |
| **Pros** | **Contras** |
| * Agrega una capa adicional de pruebas más allá de las pruebas unitarias * Más estable que las pruebas UI; tanto a la hora de desarrollar como de mantener pues hay menos cambios en esta capa * Ejecución más rápida; mucho más rápido que las pruebas de IU / E2E. | * Si bien las pruebas de integración aumentan el alcance de las pruebas, no se centran en la capa de experiencia del usuario (UI) |

Pros y contras de las pruebas de integración

Prueba de interfaz de usuario

La capa de prueba de la interfaz de usuario es importante, pero no debería constituir la mayor parte de sus pruebas. Esta capa se centra en la experiencia del usuario (UX) al garantizar que los flujos de la interfaz de usuario (UI) funcionen como se esperaba.

|  |  |
| --- | --- |
| **Pros** | **Contras** |
| * Garantiza que la experiencia del cliente esté intacta * Las herramientas de grabación y ejecución (*record and play*) pueden hacer que esta capa sea más fácil de automatizar | * Si bien son más fáciles de automatizar requieren mucho más tiempo para diseñar y automatizar * Pruebas anémicas o poco resistentes, ya que la interfaz de usuario suele cambiar con frecuencia; mayor probabilidad de reportar falsos positivos * Los tiempos de ejecución son más largos. Incluso, algunas recomendaciones son cerrar la sesión y el navegador antes de cada ejecución. |

Pros y contras de las pruebas de interfaz de usuario