Frameworks de automatización de pruebas

DigitalHouse >
Coding School

Certified Tech Developer
The Ultimate Degree

Índice

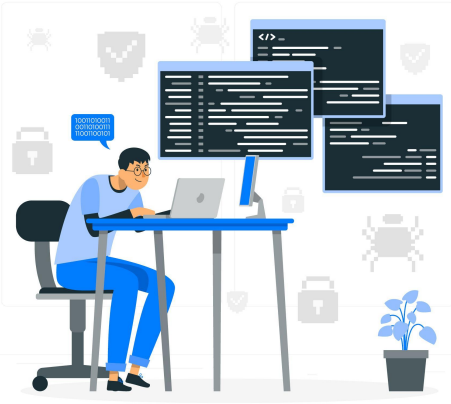
1 Generalidades

Un ***framework*** define un conjunto de reglas/pautas o mejores prácticas que podemos seguir de manera sistemática para lograr los resultados deseados.

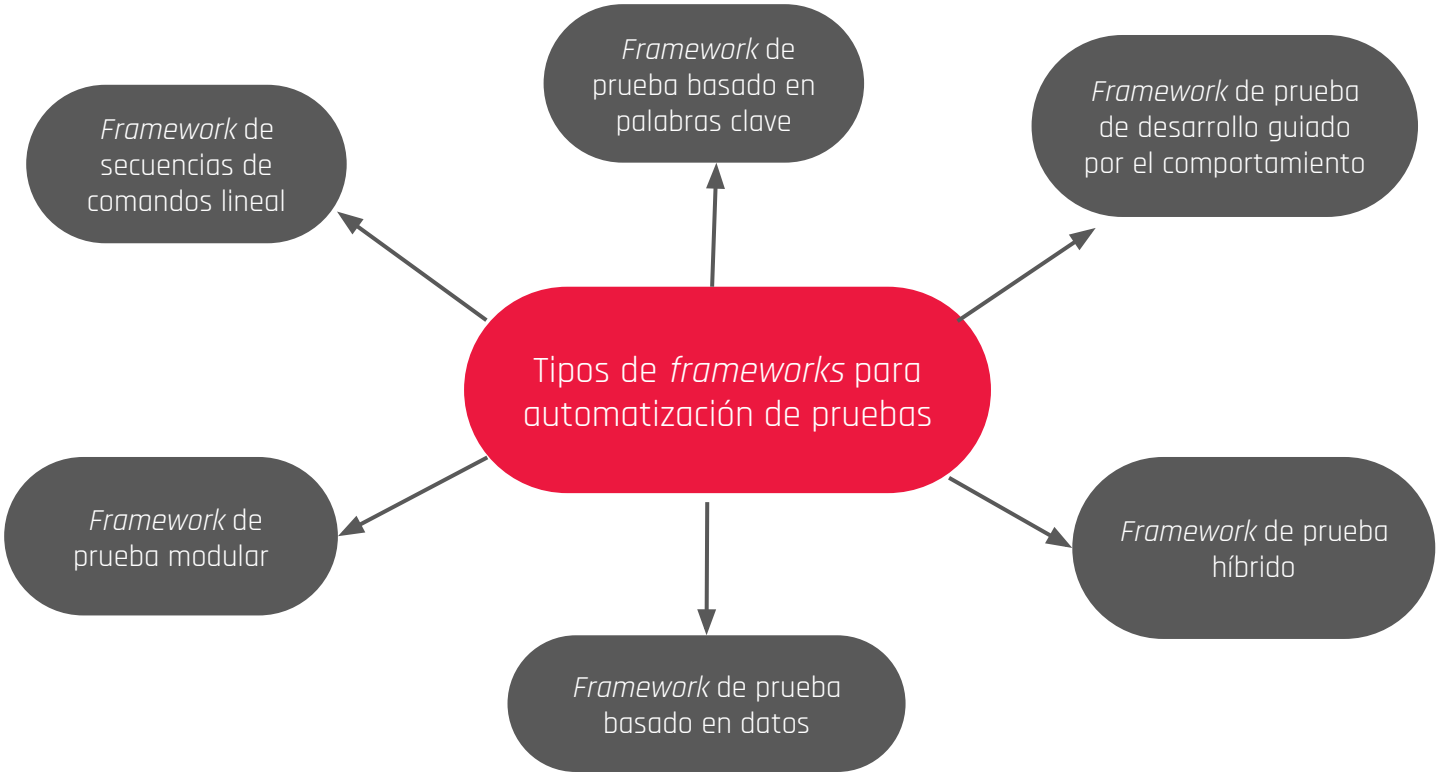
Generalidades

Un ***framework* de automatización** de pruebas es un **conjunto de pautas** como estándares de codificación, manejo de datos de prueba, tratamiento de repositorios de objetos, etc., que cuando se siguen durante la secuencia de comandos de automatización producen resultados beneficiosos. Estos pueden ser una mayor reutilización de código, mayor portabilidad, menor costo de mantenimiento de secuencias de comandos, etc.

Se trata de pautas y no reglas en el sentido estricto de la palabra ya que no son obligatorios. Se puede escribir un script sin seguir las pautas, pero se perderán las ventajas de tener un ***framework***.



2 Tipos de *frameworks*


Tipos de *frameworks* para automatización de pruebas
Framework de prueba basado en palabras clave
Framework de prueba de desarrollo guiado por el comportamiento
Framework de secuencias de comandos lineal
Framework de prueba modular
Framework de prueba híbrido
Framework de prueba basado en datos

Framework de secuencias de comandos lineal

Conocido como “grabación y reproducción” es el más simple de todos los *frameworks* de automatización de pruebas. En este *framework* el tester registra manualmente cada paso (navegación y entradas del usuario) e inserta puntos de control (pasos de validación) en la primera iteración. Luego, reproduce el script grabado en las iteraciones siguientes.

Ejemplos: Selenium GRID, UFT, Test Complete, Sahi.

TestComplete

HP UFT

Framework de secuencias de comandos lineal

- La forma más rápida de generar un script.
- No se requiere experiencia en automatización.
- La forma más sencilla de conocer las funciones de la herramienta de prueba.

- Poca reutilización de guiones.
- Los datos de prueba están codificados en el script.
- Pesadilla de mantenimiento.

Framework de prueba modular

En el *framework* de prueba modular, los probadores crean módulos de scripts de prueba dividiendo la aplicación completa bajo prueba en pruebas independientes más pequeñas.

En palabras simples, los evaluadores dividen la aplicación en varios módulos y crean scripts de prueba individualmente. Estos se pueden combinar para hacer scripts de prueba más grandes utilizando uno maestro para lograr los escenarios requeridos. Este script maestro se utiliza para invocar los módulos individuales para ejecutar escenarios de prueba de un extremo a otro (E2E).

La razón principal para usar este *framework* es construir una capa de abstracción para proteger el módulo maestro de cualquier cambio realizado en las pruebas individuales.

Framework de prueba modular

- Mejor escalabilidad y más fácil de mantener debido a la división de la aplicación completa en diferentes módulos.
- Puede escribir scripts de prueba de forma independiente.
- Los cambios en un módulo tienen un impacto nulo o bajo en los otros módulos.

- Toma más tiempo analizar los casos de prueba e identificar flujos reutilizables.
- Debido a los datos codificados en los scripts de prueba no es posible demandar varios conjuntos de datos.
- Requiere habilidades de codificación para configurar el *framework*.

Framework de prueba basado en datos

En este *framework*, mientras que la lógica del caso de prueba reside en los scripts de prueba, **los datos de prueba se separan** y se mantienen fuera de los scripts de prueba. Los datos de prueba se leen de los archivos externos (archivos de Excel, archivos de texto, archivos CSV, fuentes ODBC, objetos DAO, objetos ADO) y se cargan en las variables dentro del script de prueba. Las variables se utilizan tanto para los valores de entrada como para los valores de verificación. Los scripts de prueba se preparan utilizando Linear Scripting o Test Library Framework.

Ejemplo: UFT, Specflow (C#)

Framework de prueba basado en datos

- Los cambios en los scripts de prueba no afectan los datos de prueba.
- Los casos de prueba se pueden ejecutar con múltiples conjuntos de datos.
- Se puede ejecutar una variedad de escenarios de prueba simplemente variando los datos de prueba en el archivo de datos externo.

Se necesita más tiempo para planificar y preparar tanto los scripts de prueba como los datos de prueba.

Framework de prueba basado en palabras clave

Es conocida como prueba basada en tablas o prueba basada en palabras de acción. El desarrollo de este *framework* requiere tablas de datos y palabras clave, independientemente de la herramienta de automatización de pruebas utilizada para ejecutarlas. Las pruebas se pueden diseñar con o sin la aplicación. En una prueba basada en palabras clave, la funcionalidad de la aplicación bajo prueba se documenta en una tabla, así como en instrucciones paso a paso para cada prueba.

Aunque trabajar en este *framework* no requiere muchas habilidades de programación, la configuración inicial (implementarlo) requiere más experiencia.

Ejemplo: Robot Framework (Python).

Framework de prueba basado en palabras clave

- Proporciona una alta reutilización de código.
- Herramienta de prueba independiente.
- Aunque la aplicación cambie, los scripts de prueba no cambian.
- Las pruebas se pueden diseñar con o sin la aplicación bajo prueba.

- Dado que la inversión inicial es bastante alta, los beneficios de esto solo se pueden obtener si la aplicación es considerablemente grande y los scripts de prueba deben mantenerse durante bastantes años.
- Se requiere una gran experiencia en automatización para crear el *framework* basado en palabras clave.

Framework de prueba híbrido

El *framework* de automatización de pruebas híbridas es la combinación de dos o más *frameworks* mencionados anteriormente. Intenta aprovechar las fortalezas y los beneficios de otros *frameworks* para el entorno de prueba particular que administra. La mayoría de los equipos están construyendo este *framework* impulsado por híbridos en el mercado actual. La industria generalmente utiliza el *framework* de palabras clave en una combinación con el de descomposición de funciones.

Ejemplo: Karate-DSL, Citrus, etc.

CITRUS

Framework de prueba de desarrollo guiado por el comportamiento

El propósito de este *framework* de desarrollo guiado por el comportamiento es crear una plataforma que permita a todos (como analistas de negocios, desarrolladores, probadores, etc.) participar activamente. Requiere una mayor colaboración entre los equipos de desarrollo y prueba, pero no requiere que los usuarios estén familiarizados con un lenguaje de programación. se utiliza un lenguaje natural no técnico para crear especificaciones de prueba (Gherkin).

Algunas de las herramientas disponibles en el mercado para el desarrollo impulsado por el comportamiento son JBehave (Java) , Cucumber , Specflow (C#), Serenity, etc.

Conclusión

Estos son los beneficios generales de contar con un *framework* de prueba:

- Ayuda a reducir el riesgo y los costos.
- Mejora la eficiencia de las pruebas.
- Ayuda a reducir el costo de mantenimiento.
- Permite la reutilización de código
- Permite alcanzar la máxima cobertura de prueba.
- Maximiza la funcionalidad de la aplicación
- Ayuda a reducir la duplicación de casos de prueba
- Ayuda a mejorar la eficiencia y el rendimiento de la prueba con la automatización de la prueba.


DigitalHouse>
Coding School