GLOSARIO

Curso:
TEST AUTOMATION ENGINEER

A

Android y React Native
Plataformas para el desarrollo de aplicaciones móviles. Android es el sistema operativo de Google; React Native es un framework de JavaScript para construir apps móviles multiplataforma.

Análisis de fallos y notificaciones automáticas
Proceso de identificar errores en la ejecución de pruebas y generar alertas automáticas para su seguimiento.

Async/Await
Sintaxis de JavaScript utilizada para manejar operaciones asíncronas de forma más legible y estructurada que las promesas tradicionales.

Automatización con GitHub Actions y GitLab CI
Uso de herramientas de integración continua para ejecutar flujos de trabajo automatizados, como pruebas o despliegues, al hacer cambios en el código.

Automatización con Supertest + Jest
Frameworks de pruebas en JavaScript utilizados para validar endpoints de APIs de manera automatizada y eficaz.

Automatización de flujos E2E complejos
Implementación de pruebas End-to-End (de extremo a extremo) que simulan el comportamiento del usuario en flujos complejos dentro de una aplicación.

Automatización de tareas básicas
Proceso de simplificación de comandos repetitivos mediante scripts o herramientas como terminal o CLI.

Automatización de un flujo completo Web + API + CI/CD
Implementación integral de pruebas automáticas que cubren tanto la interfaz web como APIs, integradas en pipelines de CI/CD.

B

Buenas prácticas de programación para automatización
Conjunto de recomendaciones técnicas para escribir código limpio, reutilizable y fácil de mantener, especialmente en pruebas automatizadas.

BrowserStack, Sauce Labs y AWS Device Farm
Servicios en la nube que permiten ejecutar pruebas automatizadas en múltiples navegadores y dispositivos reales.

C

Captura de evidencia (screenshots, videos, reportes)
Registro visual de los resultados de pruebas para facilitar el análisis de errores o verificación de comportamientos esperados.

Ciclo de vida del bug y reporte de incidencias
Etapas desde la detección de un defecto hasta su resolución, incluyendo documentación, seguimiento y cierre.

CI/CD (Integración y Entrega Continua)
Práctica DevOps que automatiza la integración de código y su despliegue continuo en entornos de producción o prueba.

Colección, validaciones y ambiente de pruebas (Postman)
Conjunto de pruebas agrupadas, configuraciones y validaciones dentro de la herramienta Postman para validar APIs.

Comandos esenciales y validaciones básicas (Cypress)
Instrucciones fundamentales para interactuar con elementos web y verificar condiciones durante pruebas automatizadas con Cypress.

Comparación con Cypress (Playwright)
Análisis de diferencias y similitudes entre Playwright y Cypress en cuanto a capacidades de testing automatizado.

Componentes móviles
Partes funcionales de una app móvil, como botones, formularios o listas, que deben ser validadas en pruebas.

Construcción y ejecución de imágenes con Docker
Proceso de empaquetar una aplicación y sus dependencias en contenedores para facilitar su ejecución y prueba.

Contextos colaborativos (BDD)
Entornos de trabajo donde los equipos técnicos y no técnicos colaboran para definir pruebas utilizando lenguaje natural.

D

Definición de features y scenarios (Gherkin)
Especificaciones funcionales escritas en lenguaje natural que describen el comportamiento esperado de una aplicación.

Desestructuración (JavaScript)
Técnica de JavaScript que permite extraer valores de objetos o arrays de forma concisa.

Detox
Framework para pruebas E2E en aplicaciones móviles React Native, alternativa a Appium.

Docker
Plataforma de contenedores que permite crear, ejecutar y gestionar aplicaciones en entornos aislados.

E

Emisión de certificado oficial de finalización
Entrega de una constancia que acredita que un participante completó con éxito un curso o proceso formativo.

Entornos aislados para QA
Ambientes de prueba independientes del entorno de producción, usados para verificar calidad sin interferencias externas.

Errores (manejo de)
Técnicas para detectar, prevenir y gestionar fallos en el código durante el proceso de testing.

Escribir pruebas en lenguaje natural (BDD)
Redacción de scripts de prueba comprensibles por todos los actores del proyecto, usando lenguaje común.

Estructuras condicionales y repetitivas (JavaScript)
Elementos del lenguaje que permiten controlar el flujo del programa (if, else, for, while).

Ejecución en distintos navegadores
Capacidad de un framework de pruebas para correr test cases en múltiples motores de navegador (Chromium, Firefox, WebKit).

Exploración de herramientas como Detox
Investigación y pruebas de herramientas especializadas para testing móvil.

F

Features (BDD)
Funcionalidades específicas de una aplicación descritas en lenguaje natural.

Fixtures (Cypress)
Archivos con datos estáticos utilizados en pruebas para simular información de entrada.

Flujo de trabajo profesional (GIT)
Conjunto de buenas prácticas para colaborar con código usando ramas, pull requests y resolución de conflictos.

Funciones puras
Funciones que no dependen de variables externas y devuelven siempre el mismo resultado para los mismos argumentos.

G

Gestores de paquetes (Node.js)
Herramientas como npm o yarn para instalar dependencias y bibliotecas en proyectos JavaScript.

Git
Sistema de control de versiones distribuido para registrar y coordinar cambios en el código fuente.

GitHub
Plataforma en línea para alojar repositorios Git, colaborar y automatizar flujos de desarrollo.

Gherkin
Lenguaje estructurado usado en BDD para definir pruebas en formato Given-When-Then.

H

Hooks (Cypress)
Funciones que permiten ejecutar código antes o después de cada prueba o conjunto de pruebas.

I

Instalación de herramientas: Node.js, VS Code
Proceso de preparación del entorno de desarrollo para programación y testing.

Instalación de Appium y configuración con emuladores
Preparación del entorno para ejecutar pruebas móviles simuladas en dispositivos virtuales.

Instalación de proyectos Cypress o Playwright
Configuración inicial de un entorno de testing para ejecutar pruebas automatizadas.

Integración con pruebas E2E
Vinculación de pruebas End-to-End con herramientas, APIs o CI/CD para asegurar la validación completa del sistema.

IA aplicada al desarrollo
Uso de inteligencia artificial, como asistentes de código (ej. ChatGPT), para mejorar la productividad y la calidad en el desarrollo de software.

J

Jest
Framework de pruebas para JavaScript, comúnmente usado para pruebas unitarias y de integración.

K

Kanban
Metodología ágil basada en la visualización del flujo de trabajo y la gestión de tareas en columnas.

L

Lenguaje natural
Forma de comunicación comprensible para humanos, utilizada en pruebas BDD para incluir a no técnicos.

M

Manejo de múltiples entornos y pruebas condicionales
Capacidad de una prueba para adaptarse a diferentes configuraciones y ejecutar solo bajo ciertas condiciones**.**

Metodologías ágiles (Scrum, Kanban)
Enfoques iterativos y colaborativos para la gestión de proyectos de desarrollo.

Mockeo de API
Simulación del comportamiento de una API para pruebas cuando el backend no está disponible.

N

Node.js
Entorno de ejecución para JavaScript en el servidor, clave en desarrollo web y automatización.

O

Objetos y Arrays (JavaScript)
Estructuras de datos clave en JavaScript que permiten almacenar y manipular información.

Organización y documentación del repositorio
Estructura del proyecto, incluyendo carpetas, convenciones, README y otros archivos informativos.

P

Page Object Model (POM)
Patrón de diseño que separa la lógica de pruebas de la representación de elementos de la interfaz, mejorando el mantenimiento del código.

Paralelismo y asincronía (Playwright)
Capacidad de ejecutar múltiples pruebas simultáneamente y manejar respuestas asíncronas.

Pipeline (Test + Build + Deploy)
Secuencia automatizada de procesos en CI/CD que prueba, compila y despliega el software.

Postman
Herramienta para desarrollar, probar y documentar APIs REST mediante colecciones y entornos de pruebas.

Promesas (JavaScript)
Objetos que representan una operación asíncrona y su resultado eventual.

Pruebas autenticadas
Verificación de funcionalidades de sistemas que requieren login o tokens de acceso.

Pruebas condicionales
Tests que se ejecutan solo si se cumplen ciertas condiciones del entorno o configuración.

Pruebas E2E (End-to-End)
Validaciones automatizadas que cubren todo el recorrido del usuario en una aplicación.

Q

QA funcional y no funcional
Pruebas que verifican si una app cumple con los requisitos funcionales (lo que debe hacer) y no funcionales (rendimiento, seguridad, etc.).

R

Repositorios
Ubicación central donde se almacena el código fuente y su historial de versiones.

Rúbrica de evaluación técnica
Instrumento para calificar el desempeño técnico de un proyecto o prueba.

S

Sanity y Smoke Testing
Pruebas rápidas para verificar que las funciones básicas del software funcionan tras un cambio (Sanity) o antes de realizar pruebas más profundas (Smoke).

Scrum
Metodología ágil que divide los proyectos en sprints y promueve la entrega iterativa de valor.

Simulación de gestos e interacciones
Reproducción automática de toques, deslizamientos y otros gestos para probar apps móviles.

Sintaxis básica y arquitectura (Playwright)
Estructura y forma de escribir scripts de prueba con el framework Playwright.

Supertest
Librería de Node.js que permite realizar pruebas HTTP sobre APIs REST.

T

Terminal
Interfaz de línea de comandos que permite ejecutar scripts y comandos para gestionar el entorno de desarrollo.

Testing en entornos móviles
Proceso de validar funcionalidades en aplicaciones diseñadas para teléfonos y tablets.

Tipos de datos (JavaScript)
Clasificaciones de valores como string, number, boolean, null, undefined, etc.

Tipos de pruebas (unitarias, integración, regresión)
Clasificación según el nivel y propósito de los tests: Unitarias (funciones aisladas), Integración (componentes combinados), Regresión (verificación tras cambios).

U

Uso colaborativo de repositorios
Trabajo en equipo sobre el mismo código fuente mediante ramas, pull requests y revisiones.

Uso de hooks personalizados
Extensiones que permiten configurar comportamientos antes o después de ejecutar pruebas.

V

Validación de errores
Comprobación de que el sistema maneja correctamente las condiciones anómalas o inesperadas.

Variables de entorno
Valores configurables usados para adaptar una app a diferentes entornos (desarrollo, testing, producción).

VS Code
Editor de código fuente liviano, extensible y popular en el desarrollo web y automatización.

Video de presentación del proyecto
Material audiovisual donde se muestra el funcionamiento de un proyecto o solución desarrollada.

W

Webkit, Chromium y Firefox
Motores de navegador utilizados para ejecutar y renderizar contenido web, compatibles con herramientas de testing como Playwright.