TEST AUTOMATION ENGINEER – FORMACIÓN INTEGRAL

{desafío} latam_

CURSO:

TEST AUTOMATION ENGINEER – FORMACIÓN INTEGRAL

- Módulo 1: FUNDAMENTOS DEL TESTING DE SOFTWARE
- Módulo 2: CONTROL DE VERSIONES Y ENTORNOS DE DESARROLLO
- **Módulo 3: FUNDAMENTOS DE PROGRAMACIÓN APLICADOS AL TESTING (JAVASCRIPT)**
- Módulo 4: CARACTERISTICAS DE CYPRESS – AUTOMATIZACIÓN WEB I
- Módulo 5: CARACTERISTICAS DE CYPRESS – AUTOMATIZACIÓN WEB II
- Módulo 6: PRUEBAS DE APIS CON POSTMAN Y SUPERTEST

Te encuentras aquí

{desafío} latam_

CURSO:

TEST AUTOMATION ENGINEER – FORMACIÓN INTEGRAL

- Módulo 7: HERRAMIENTAS DE PLAYWRIGHT Y PRUEBAS CON MÚLTIPLES NAVEGADORES
- Módulo 8: DESARROLLO GUIADO EN EL COMPORTAMIENTO (BDD) CON CUCUMBER.JS
- Módulo 9: HERRAMIENTAS DE AUTOMATIZACIÓN MÓVIL CON APPIUM
- Módulo 10: HERRAMIENTAS DE INTEGRACIÓN DE PRUEBAS EN CI/CD
- Módulo 11: HERRAMIENTAS DE DOCKER, ENTORNOS VIRTUALIZADOS Y PRUEBAS EN LA NUBE
- HERRAMIENTAS DE AUTOMATIZACIÓN DE UN FLUJO COMPLETO WEB + API + CI/CD

{desafío} latam_

Módulo 3: FUNDAMENTOS DE PROGRAMACIÓN APLICADOS AL TESTING (JAVASCRIPT).

{desafío} latam_

OBJETIVO ESPECÍFICO DEL MÓDULO

- DISTINGUIR LOS FUNDAMENTOS DE PROGRAMACIÓN APLICADOS AL TESTING (JAVASCRIPT), DE ACUERDO A LAS APLICACIONES WEB, MÓVILES Y APIS.

{desafío} latam_

¿Por qué cree que es importante comprender conceptos básicos de programación —como tipos de datos, funciones y estructuras— al momento de automatizar pruebas de software?

{desafío} latam_

TIPOS DE DATOS Y ESTRUCTURAS DE CONTROL

- **Tipos de datos en JavaScript:**
- **Primitivos:** string, number, boolean, null, undefined
- **Estructurados:** object, array
- **Funciones:**
- **Declarativas:** function miFuncion() {...}

- **Expresivas:** const miFuncion = () => {...}

- **Estructuras condicionales:**
- if, else, else if, switch
- Estructuras repetitivas:
- for, while, do...while, forEach, map

- Estos elementos permiten construir scripts de validación, generación de datos o automatización de rutinas simples.

OBJETOS, ARRAYS Y CONTROL DE ERRORES

- **Objetos y arrays:**

- Claves y valores ({ nombre: "QA", edad: 30 })
- Listas indexadas ([1, 2, 3])

- **Desestructuración:**
- const {nombre} = objeto;
- const [primero, segundo] = array;

{desafío} latam_

- **Manejo de errores:**
- try { ... } catch (error) { ... }
- Permite capturar excepciones sin que el programa se detenga.
- Útil para pruebas automatizadas donde el error debe ser manejado y reportado.

PROGRAMACIÓN ASINCRÓNICA Y FUNCIONES PURAS

- **Funciones puras:**
- Siempre devuelven el mismo resultado con los mismos parámetros.
- No modifican datos externos (sin efectos secundarios).
- Facilitan testeo automatizado.

- **Promesas:**
- Manejo de procesos asincrónicos: new Promise((resolve, reject) => {...})

{desafío} latam_

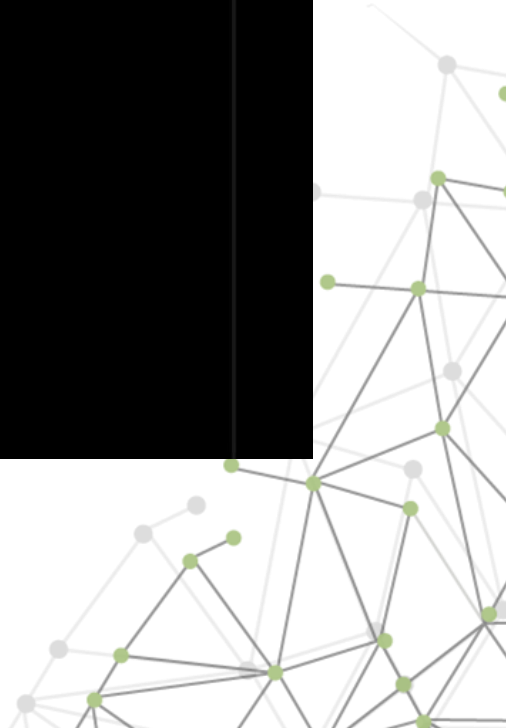
INSTALACIÓN DE HERRAMIENTAS ESENCIALES

- **Node.js:**
- Plataforma para ejecutar JS fuera del navegador, indispensable para entornos de automatización QA modernos.
- Requiere instalar desde https://nodejs.org

- **VS Code:**
- Editor de código ligero y extensible.
- Recomendado para QA por sus extensiones útiles (GIT, Jest, ESLint, Playwright).

- **async/await:**
- Sintaxis moderna para manejar promesas de forma legible:

```
async function obtenerDatos() {
  try {
    const res = await fetch(url);
    const data = await res.json();
  } catch (e) {
    console.error(e);
  }
}
```



{desafío} latam_

BUENAS PRÁCTICAS DE CÓDIGO EN AUTOMATIZACIÓN QA

- **Clareza:** nombres de variables y funciones descriptivas (getUserEmail)
- **Reutilización:** dividir el código en funciones reutilizables
- **DRY (Don’t Repeat Yourself):** evitar duplicación de lógica
- **Modularización:** separar funcionalidades por archivos o módulos

{desafío} latam_

- **Testing desde el diseño:** escribir el código pensando en cómo será testeado
- **Comentarios útiles:** explicar por qué se hace algo, no qué se hace

Estas prácticas mejoran la mantenibilidad, reducen errores y facilitan el trabajo en equipo.

USO DE INTELIGENCIA ARTIFICIAL EN LA PROGRAMACIÓN

- **Asistentes como ChatGPT pueden:**
- Sugerir código o corregir errores.
- Generar scripts de pruebas automatizadas.
- Explicar código difícil de entender.
- Crear funciones a partir de descripciones

{desafío} latam_

- **Ventajas:**
- Acelera el desarrollo.
- Reduce errores lógicos o de sintaxis.
- Mejora la productividad individual del QA Automation.

- **Buenas prácticas de uso:**
- Validar siempre el código sugerido.
- Utilizarlo como apoyo, no como reemplazo del razonamiento lógico.
- Documentar las decisiones tomadas con IA.

Este Módulo tiene 2 ejercicios prácticos que usted debe desarrollar…

{desafío} latam_

¿Cómo podría aplicar los principios de buenas prácticas de programación y el uso de asistentes de IA como ChatGPT para optimizar su trabajo en tareas de automatización de pruebas?

{desafío} latam_

Éxito en la evaluación parcial y en la Prueba Final...

{desafío}
latam_

Academia de
talentos digitales