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 6: Pruebas de apis con postman y supertest.

{desafío} latam_

OBJETIVO ESPECÍFICO DEL MÓDULO

- EMPLEAR PRUEBAS DE APIS CON POSTMAN Y SUPERTEST, DE ACUERDO A LAS APLICACIONES WEB, MÓVILES Y APIS.

{desafío} latam_

¿Por qué cree que es importante validar el comportamiento de una API de forma automatizada, y qué ventajas podría tener en comparación con una prueba manual?

FUNDAMENTOS DE REST Y FORMATO JSON

- **REST (Representational State Transfer):**
- Arquitectura de servicios web basada en recursos y operaciones HTTP.

- **Métodos comunes:** GET, POST, PUT, DELETE, PATCH

- **Estructura típica:**
- **Endpoint:** https://api.ejemplo.com/usuarios
- **Métodos:** GET para obtener, POST para crear, etc.

- **Formato JSON:**
- Estándar para enviar y recibir datos.

{desafío} latam_

- **Ejemplo:**

```
{
  "nombre": "Catalina",
  "email": "cata@example.com"
}
```

- Comprender el comportamiento de las APIs es crucial para realizar pruebas funcionales, validar flujos, y automatizar el backend.

POSTMAN PARA PRUEBAS MANUALES DE APIS

- **Colecciones:** agrupan solicitudes por módulo o flujo de negocio.
- **Variables de entorno:** permiten cambiar fácilmente entre entornos ({{base_url}})
- **Validaciones en tests:**
- **Usan JavaScript en la pestaña “Tests”:**

```
pm.test("Status 200", () => {
  pm.response.to.have.status(200);
});
```

- **Ambientes comunes:**
- dev, QA, staging, prod

- **Exportación y documentación:**
- Postman permite generar documentación automática y compartir colecciones por equipo

- Es ideal para validar respuestas antes de automatizar y para integrarse a CI/CD vía Newman.

AUTOMATIZACIÓN DE APIS CON SUPERTEST Y JEST

- **Supertest:** librería que permite simular llamadas HTTP en Node.js
- **Jest:** framework de testing con capacidades de assertions, mocks y reporter
- **Instalación:**

```
npm install supertest jest
```

{desafío} latam_

- **Ejemplo de prueba:**

```
const request = require('supertest');
const app = require('../app');

describe('GET /usuarios', () => {
  it('debe retornar status 200', async () => {
    const res = await request(app).get('/usuari
    expect(res.statusCode).toBe(200);
  });
});
```

- Esta combinación permite automatizar validaciones de forma robusta, rápida y repetible en pipelines de CI.

{desafío} latam_

PRUEBAS AUTENTICADAS Y MANEJO DE ERRORES

- **Autenticación:**
- Uso de tokens JWT o API keys.

- **Se envían en los headers:**

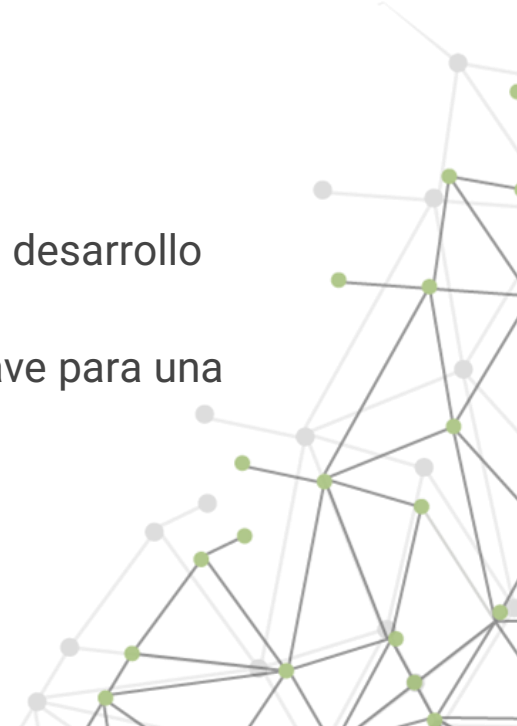
```
request(app)
  .get('/datos')
  .set('Authorization', `Bearer ${token}`)
```

- **Pruebas negativas:**

- **Validar que el sistema responde correctamente ante errores:**
- Status 401 si no hay token
- Status 400 ante datos inválidos
- Mensajes de error esperados

- **Mockeo de usuarios autenticados:**
- Permite simular un flujo completo sin usar credenciales reales en desarrollo

- Validar tanto lo que debe pasar como lo que no debe pasar es clave para una cobertura efectiva.



INTEGRACIÓN DE PRUEBAS E2E CON UI Y API

- **Flujo completo de validación:**
- Cypress automatiza el frontend
- Supertest/Jest validan la lógica del backend

- **Estrategia de integración:**
- Crear usuarios vía API → validar que aparezcan en el UI
- Ingresar datos desde la interfaz → validar con llamado API

{desafío} latam_

- **Herramientas de CI (integración continua):**
- GitHub Actions, GitLab CI, Jenkins permiten ejecutar ambas capas de prueba en cada push

- La automatización E2E asegura que tanto interfaz como lógica del negocio están funcionando correctamente de forma conjunta.

No olvide desarrollar los ejercicios que contiene el Módulo…

{desafío} latam_

¿Cómo complementan las pruebas de APIs —incluyendo validaciones, autenticaciones e integración con pruebas E2E— el aseguramiento de calidad de una aplicación desde una perspectiva integral?

{desafío} latam_

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

{desafío}
latam_

Academia de
talentos digitales