



Web Testing

"Codifica como si la persona que mantendrá tu código fuera un psicópata violento que sabe dónde vives."

-- 🖊️ Martin Golding



Objetivos

- Conocer la **terminología** y la filosofía de los distintos tipos de pruebas.
- Comprobar funcionalidades con pruebas ***end to end*** de aplicaciones web.
- Refactorizar código ***legacy*** con la tranquilidad de las pruebas unitarias.
- Crear nuevo código bajo el paradigma ***Test Driven Development***.



A quién va dirigido

- Programadores de aplicaciones con experiencia.
- Conocimientos de tecnología web: HTML y JavaScript.




Material necesario

- Editor de código y navegador modernos.
 - Recomendados *VSCode* y *Chrome*
- *Node* versión 12
- Capacidad para instalar paquetes desde *npm*

Introducción

- Las pruebas del software han sido **ignoradas y hasta despreciadas** por muchos.
- Otros las ven como una *absurda obligación*.
- No más *negacionismo*!
 - Hay relación directa entre **calidad y pruebas**.

- ✓ La detección temprana de **errores**,
 - ✓ la validación de las **funcionalidades**
 - ✓ la mejora en el **diseño** del código.
-  **Las pruebas ahorran dinero.**



Hacer pruebas

Proceso

- Aprendizaje
- Adopción

En este curso te mostraré

- Los fundamentos
- Las técnicas

Al finalizar podrás:

- Incluirlos inmediatamente
- **Mejorar la calidad de tus programas.**

Herramientas

JavaScript Tienes todo lo necesario. Puedes hacer pruebas sin framework.

Puppeteer Manipular el navegador. Ideal para e2e no funcional

Lighthouse comprobación de rendimiento, SEO...

Cypress pruebas funcionales de integración o e2e

Jest *zero configuration*. Ideal para *unit testing* y *TDD*

✅ 0 - TEST Software que funciona.

↻ Tipos de test

👨🏫 +2 Filosofía y patrones

✅ 1 -Introducción práctica a las pruebas.

🧪 Pruebas de funciones puras

🧪 Pruebas de integración de clases

🧪 Pruebas unitarias

🧪 Código TDD

2 - E2E: Pruebas web de principio a fin.

 Pruebas de contenido y visualización con Puppeteer

 Pruebas de rendimiento web con Lighthouse

3 - Pruebas funcionales con Cypress

 Pruebas de comportamiento

 Automatización e integración continua

 4 - Pruebas unitarias con Jest

 Pruebas con espías y dobles

 Pruebas de código asíncrono

5 - TDD: desarrollo guiado por las pruebas

 El ciclo virtuoso.

 Mejores resultados y mejor diseño.

"Los desarrolladores no tienen que justificar las **pruebas** y la refactorización; porque esas disciplinas aumentan su eficiencia y su **productividad**."

-- 🖊️ Robert C. Martin

 Alberto Basalo

2020