TEMA ASIGNADO

implementación de testing automatizado mediante la aplicación de pruebas unitarias y pruebas de integración, utilizando como caso práctico el proyecto 3d-print-pricer.

El objetivo principal no es el desarrollo del sistema de cálculo de impresión 3D en sí, sino la incorporación de una estrategia de aseguramiento de calidad que permita validar de manera automática el correcto funcionamiento del software.

PROBLEMA QUE INTENTA RESOLVER

En el desarrollo de software, uno de los principales desafíos es garantizar que el sistema continúe funcionando correctamente a medida que evoluciona. En ausencia de pruebas automatizadas, la verificación del funcionamiento depende de pruebas manuales, las cuales presentan diversas limitaciones:

- No garantizan cobertura total del sistema.
- Son repetitivas y propensas a errores humanos.
- No detectan automáticamente regresiones ante modificaciones del código.
- Incrementan el costo de mantenimiento a largo plazo.

En el caso del proyecto 3d-print-pricer, cuya funcionalidad principal consiste en calcular presupuestos de impresión 3D en base a múltiples parámetros (material, tiempo de impresión, costos fijos y margen de ganancia), cualquier error en la lógica de cálculo podría generar resultados incorrectos.

Por lo tanto, el problema central abordado en este trabajo es:

> ¿Cómo garantizar, de manera sistemática y automatizada, que la lógica de negocio y la integración de los distintos componentes del sistema funcionen correctamente ante cambios o ampliaciones futuras?

La solución propuesta consiste en la implementación de:

- **Pruebas Unitarias**, para validar funciones individuales de manera aislada.
- **Pruebas de Integración**, para verificar el funcionamiento conjunto de los módulos del sistema.

ARQUITECTURA DE LA SOLUCIÓN

la estrategia de testing implementada se estructura en dos niveles complementarios:

Pruebas Unitarias

Las pruebas unitarias tienen como objetivo verificar el correcto funcionamiento de unidades mínimas de código, generalmente funciones individuales.

En el caso del proyecto analizado, se testearon funciones tales como:

- Cálculo del costo de material.
- Cálculo del costo por tiempo de impresión.
- Aplicación del margen de ganancia.
- Cálculo del precio final.
- Validaciones de datos de entrada.

Cada prueba unitaria:

- Evalúa una función de manera aislada.
- No depende de la interfaz de usuario.
- No interactúa con otros módulos del sistema.
- Compara el resultado obtenido con el resultado esperado.

Esto permite detectar errores específicos en la lógica interna sin interferencia de otros componentes.

Pruebas de Integración

Las pruebas de integración verifican la correcta interacción entre distintos módulos del sistema.

En este caso, se validó:

- El flujo completo desde la entrada de datos hasta la obtención del precio final.
- La interacción entre las funciones de cálculo parciales.
- La coherencia del resultado global ante distintos escenarios.

A diferencia de las pruebas unitarias, estas pruebas no evalúan funciones aisladas, sino que analizan el comportamiento del sistema como una unidad funcional.

Esto permite detectar errores derivados de la interacción entre componentes, que podrían no manifestarse en pruebas individuales.

TECNOLOGÍAS UTILIZADAS

La pagina esta Hosteada localmente en windows 11 usando terminales de WSL (windows subsystem for linux) con una instalación de Ubuntu 24.04.1LTS
Usa:
monorepo Manager pnpm
Runtime: Node.js
Frontend: React 18 + Vite
Backend: Fastify
Lenguaje: TypeScript
Testing automatizado : Vitest y Zod
Para iniciarlo se hace por consola desde la raiz del proyecto con
pnpm dev
y para hacer los test individuales se hacen con
pnpm --filter @app/core test
pnpm --filter @app/api test
pnpm --filter @app/web test
y si se ejecuta por separado para levantar el proyecto de forma local se hace con 3 terminales:
terminal 1
pnpm --filter @app/core dev
terminal 2
pnpm --filter @app/api dev
terminal 3
pnpm --filter @app/web dev

PRINCIPALES DIFICULTADES ENCONTRADAS

Comprensión conceptual de los niveles de testing
Diferenciar claramente entre pruebas unitarias y pruebas de integración implicó analizar el alcance y objetivo de cada una.

Gestión de dependencias internas
Algunas funciones dependían de valores externos, lo cual obligó a reorganizar el código para hacerlo más testeable.
Otra dificultad es entender el concepto de cómo se implementan los test, ya que al pasar el test los cambios quedan como si se “implementaran”.