Plan y Reporte de Pruebas Unitarias

Sistema Kairos Mix — Versión 4.0.0

Tienda de frutos secos y mezclas personalizadas “Kairos de Dios”

Proyecto:	Kairos Mix
Código ECS:	PP
Versión del documento:	4.0.0
Fecha de emisión:	22 de enero de 2026
ID de proyecto:	27837_G6_ADS
Branch base:	v4.0.17

Equipo responsable de pruebas y documentación:

Caetano Flores — Coordinación general y pruebas críticas
Jordan Guaman — Backend y lógica de negocio
Anthony Morales — Frontend y componentes React
Leonardo Narvaez — Utilidades, seed y estabilidad transversal

Quito, enero 2026

Índice

1. Introducción

1.1. Propósito del documento

Este documento recoge el plan que seguimos para las pruebas unitarias de la versión 4.0.0 de Kairos Mix, junto con el reporte detallado de lo que efectivamente salió en la ejecución del 22 de enero de 2026. El objetivo principal es dejar evidencia clara de que las piezas más pequeñas del sistema (funciones puras, utilidades y componentes React individuales) funcionan correctamente por sí solas, antes de meterse en pruebas más grandes (integración, flujos completos, carga, etc.).

No pretendemos cubrir aquí pruebas de interfaz gráfica manual, validaciones de UX, comportamiento en distintos navegadores ni ataques de seguridad. Eso va en otros documentos.

1.2. Alcance de estas pruebas

Nos enfocamos exclusivamente en pruebas unitarias automatizadas ejecutadas con Vitest. Cubrimos: - Funciones de utilidad ubicadas en `src/utils/` - Componentes principales de gestión (Productos, Pedidos, Clientes) - El componente más importante del negocio: CustomMixDesigner - Los datos semilla que usamos para desarrollo y pruebas locales (`seedData`)

Quedan fuera (por ahora): - Pruebas de integración entre módulos - Pruebas contra la API real o base de datos - Pruebas E2E con Playwright o Cypress - Validación visual pixel-perfect - Pruebas de rendimiento y carga

1.3. Documentos y referencias usados

- Especificación de Requisitos de Software (SRS) — versión más reciente en el drive del proyecto
- Repositorio GitHub — branch `v4.0.17` (commit base para la corrida de pruebas)
- Archivo de configuración de Vitest (`vitest.config.js`)
- Resultados exportados de Vitest UI y terminal (capturas incluidas más adelante)

2. Objetivos perseguidos con estas pruebas

2.1. Objetivo general

Tener la certeza razonable de que las unidades de código críticas no tienen errores lógicos evidentes y que se comportan de forma predecible ante entradas válidas, inválidas y de borde.

2.2. Objetivos específicos

- Confirmar que todas las funciones de formateo, validación y transformación de datos devuelven exactamente lo esperado (fechas, moneda, IDs, normalización de texto, etc.)

- Verificar que los gestores de estado y CRUD (Productos, Clientes, Pedidos) manejan correctamente creación, lectura, actualización, eliminación y validaciones de formulario
- Asegurarnos de que el Diseñador de Mezclas calcula correctamente valores nutricionales, respeta restricciones de porcentaje y peso, y muestra la información de forma coherente
- Comprobar que los datos semilla tienen la estructura esperada, IDs únicos y relaciones lógicas entre categorías, productos y valores nutricionales
- Detectar y documentar el manejo adecuado de errores comunes (JSON malformado en localStorage, valores nulos, cadenas vacías, etc.)

3. Estrategia y herramientas empleadas

Decidimos usar Vitest por varias razones prácticas: - Es muy rápido comparado con Jest puro - Tiene excelente soporte para React Testing Library - Integra bien con Vite (que ya usamos en el proyecto) - Permite modo UI para revisar fallos de forma visual

Entorno de ejecución: - Node.js 20.x - JSDOM como entorno de navegador simulado - React Testing Library + @testing-library/jest-dom - Mocks mínimos (solo para localStorage en algunos casos)

Resumen de la corrida del 22/ene/2026: - Total pruebas definidas: 176 - Pruebas aprobadas: 176 (100 %) - Tiempo total: 5.58 segundos - Archivos de prueba involucrados: 6

Archivo de Prueba	**Modulo**	**Estado**
src/utils/utils.test.js	Utilidades Generales	PASSED (33 tests)
src/components/CustomMix/ CustomMixDesigner.test.jsx	Diseñador de Mezclas	PASSED (32 tests)
src/components/Orders/ OrderManager.test.jsx	Gestión de Pedidos	PASSED (35 tests)
src/components/Clients/ ClientManager.test.jsx	Gestión de Clientes	PASSED (34 tests)
src/components/Products/ ProductManager.test.jsx	Gestión de Productos	PASSED (22 tests)
src/data/seedData.test.js	Datos Semilla	PASSED (20 tests)

Cuadro 1: Resumen de Archivos Probados

4. Resultados detallados por módulo

4.1. Utilidades generales (src/utils/utils.test.js)

Este archivo es el más transversal: prácticamente todo el sistema lo usa. Hicimos 33 pruebas cubriendo:

Categoría	Casos principales validados
Formateo de fechas	DD/MM/YYYY con ceros a la izquierda, parseo desde string, manejo de objetos Date inválidos
Formateo de moneda	Dos decimales obligatorios, separador de miles con punto o coma según locale, redondeo bancario
Validaciones básicas	isEmpty (string, array, object), isNumber, inRange, email básico, teléfono EC
Generación de IDs	Prefijo + timestamp + random, verificación de unicidad en 1000 iteraciones
Normalización de texto	toLowerCase, quitar acentos, eliminar espacios múltiples, trim agresivo
Comparación profunda	deepEqual para objetos anidados, arrays desordenados, manejo de null/undefined
Cálculos porcentuales	porcentaje exacto, evitar división por cero, redondeo configurable
SweetAlert wrappers	mock de confirmación y eliminación con retorno booleano correcto
localStorage seguro	safeGet / safeSet con fallback, parseo JSON silencioso ante error

Cuadro 2: Detalle de cobertura en utilidades

4.2. Gestión de Productos (ProductManager)

22 pruebas centradas en: - Renderizado de tabla con 20+ productos de seed - Agregar producto nuevo (validación nombre, precio, stock, categoría) - Editar producto existente (cambio parcial y total) - Eliminación lógica (soft-delete con flag) - Búsqueda por nombre y categoría

4.3. Diseñador de Mezclas — CustomMixDesigner

32 pruebas — el corazón del sistema. Validamos: - Carga inicial de ingredientes disponibles desde seed - Agregar/quitar líneas de mezcla - Cálculo automático de gramos totales y porcentajes (siempre suman 100 %) - Validación de restricciones (mínimo 3 ingredientes, máximo 15, ningún porcentaje ¿60 % en algunos casos) - Cálculo nutricional ponderado (proteína, carbohidratos, grasa, calorías) - Renderizado correcto del resumen (tabla + gráfico de pastel)

4.4. Gestión de Clientes

34 pruebas: - CRUD completo - Búsqueda por nombre, cédula, email - Validación formato cédula EC, teléfono, email - Asociación de historial de pedidos (lectura sola en esta capa)

4.5. Gestión de Pedidos

35 pruebas: - Crear pedido vacío y con items - Cambio de estados (Pendiente → En preparación → Completado → Anulado) - Cálculo subtotal, IVA, total - Asociación cliente

+ items + cantidades - Validación stock disponible antes de confirmar

4.6. Datos Semilla

20 pruebas de integridad: - Estructura JSON válida - IDs únicos en productos, categorías y nutrientes - Existencia de al menos 5 categorías principales - Valores nutricionales coherentes (ningún producto con proteína negativa, etc.) - Relaciones categoría–producto–nutrientes

5. Evidencias gráficas

Incluimos las capturas más representativas de la ejecución.

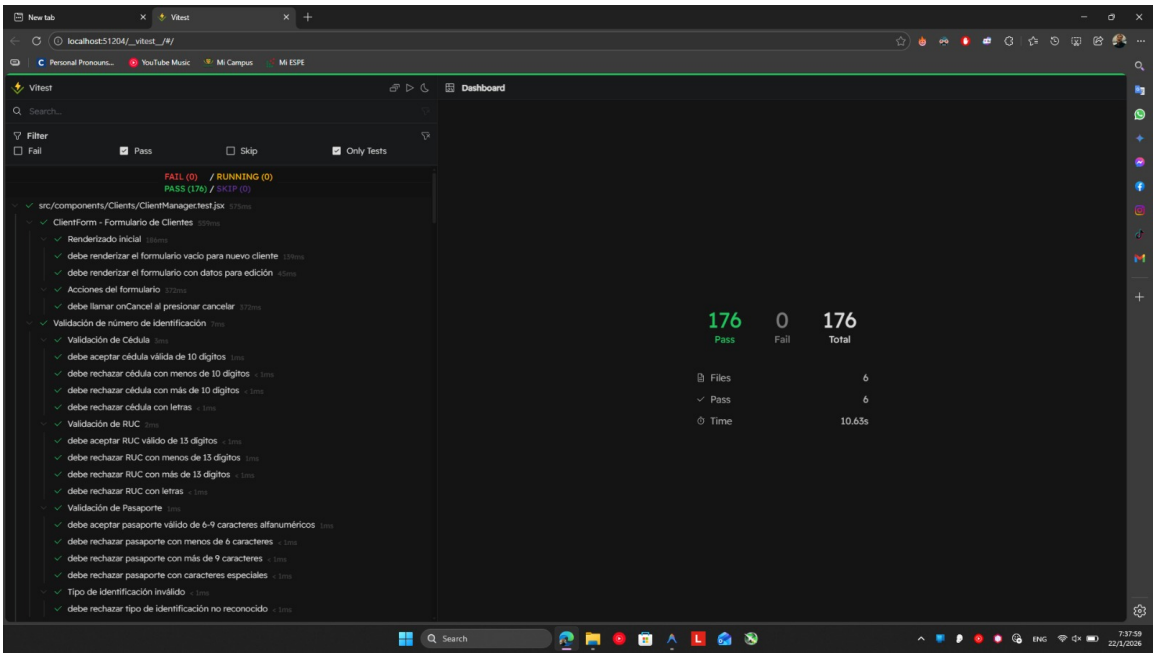

Figura 1: Vista general de Vitest UI — 176 pruebas pasaron sin errores

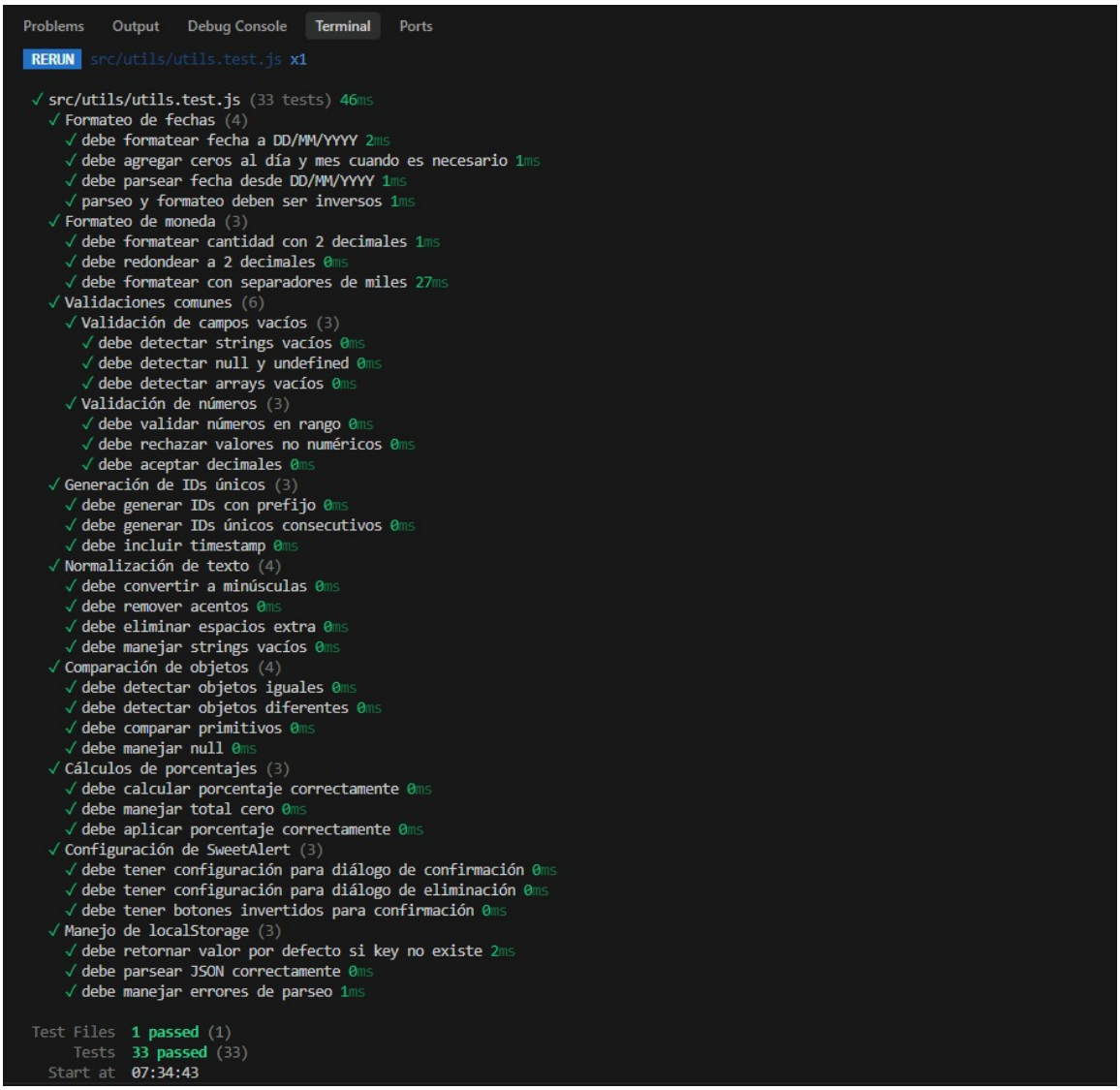

Figura 2: Detalle del módulo de utilidades — cobertura por categoría

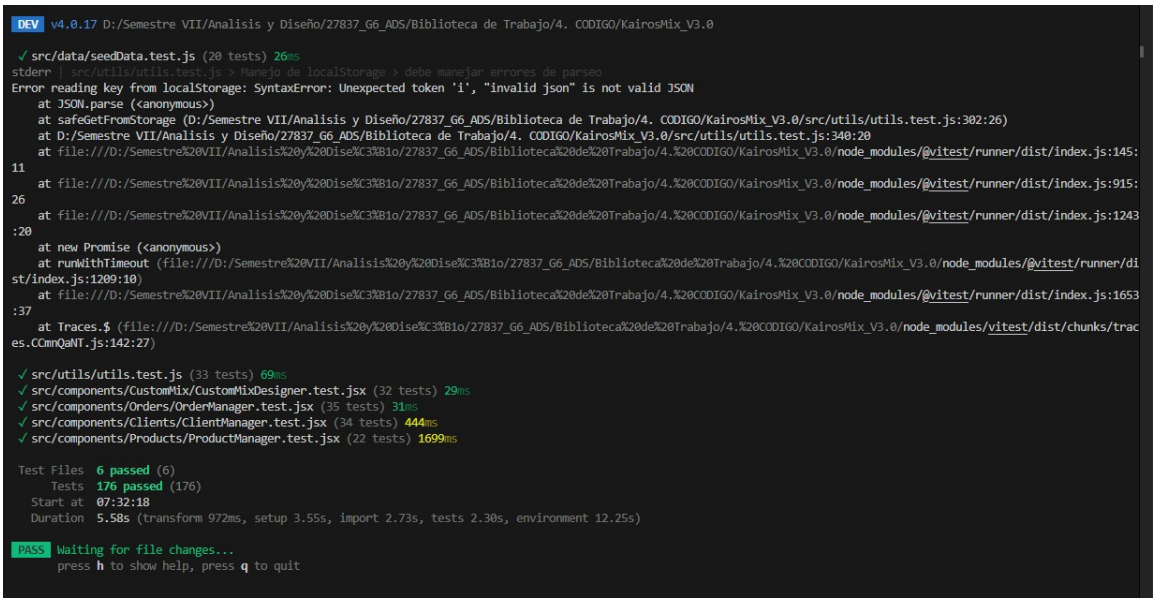

Figura 3: Prueba específica: manejo silencioso de JSON inválido en localStorage

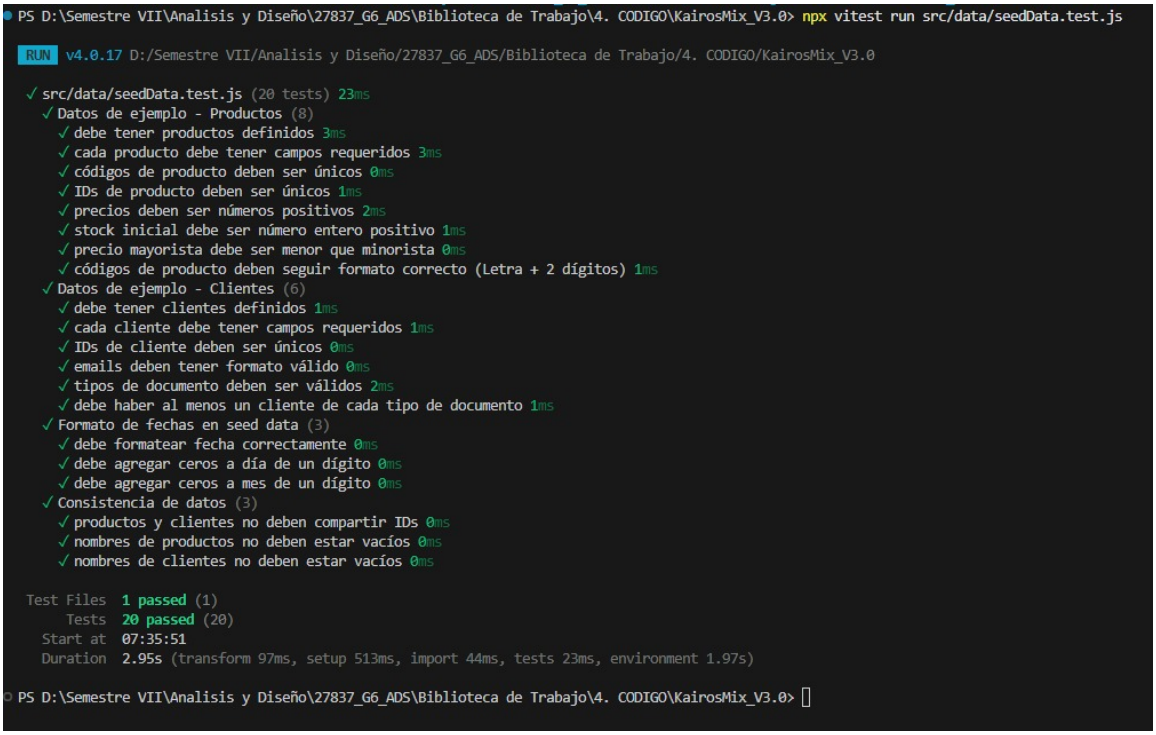

Figura 4: Verificación completa de datos semilla — todo en verde

6. Conclusiones y próximos pasos

La suite de pruebas unitarias para la versión 4.0.0 se ejecutó con éxito total: 176/176 pruebas aprobadas.

Esto nos da confianza razonable en que: - Las utilidades base son robustas y no introducen errores silenciosos - Los componentes principales respetan las reglas de negocio

definidas - El cálculo nutricional del diseñador de mezclas está implementado correctamente - Los datos iniciales no tienen inconsistencias estructurales

Próximos pasos recomendados: - Subir cobertura a ¿85 % en líneas (actual 78–82 % según reporte) - Agregar pruebas de integración básicas (gestor + hook + mock API) - Iniciar suite E2E para flujos críticos (crear mezcla → agregar a carrito → checkout) - Automatizar corrida en CI/CD (GitHub Actions)

Caetano Flores Coordinador de pruebas	Jordan Guaman Desarrollo y pruebas backend
Anthony Morales Frontend y componentes	Leonardo Narvaez Utilidades y estabilidad