

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	2
1.1. Propósito del documento	2
1.2. Alcance de estas pruebas	2
1.3. Documentos y referencias usados	2
2. Objetivos perseguidos con estas pruebas	2
2.1. Objetivo general	2
2.2. Objetivos específicos	2
3. Estrategia y herramientas empleadas	3
4. Resultados detallados por módulo	3
4.1. Utilidades generales (src/utls/utls.test.js)	3
4.2. Gestión de Productos (ProductManager)	4
4.3. Diseñador de Mezclas — CustomMixDesigner	4
4.4. Gestión de Clientes	4
4.5. Gestión de Pedidos	5
4.6. Datos Semilla	5
5. Evidencias gráficas	5
6. Conclusiones y próximos pasos	7

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/utills/utills.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/utills/utills.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.

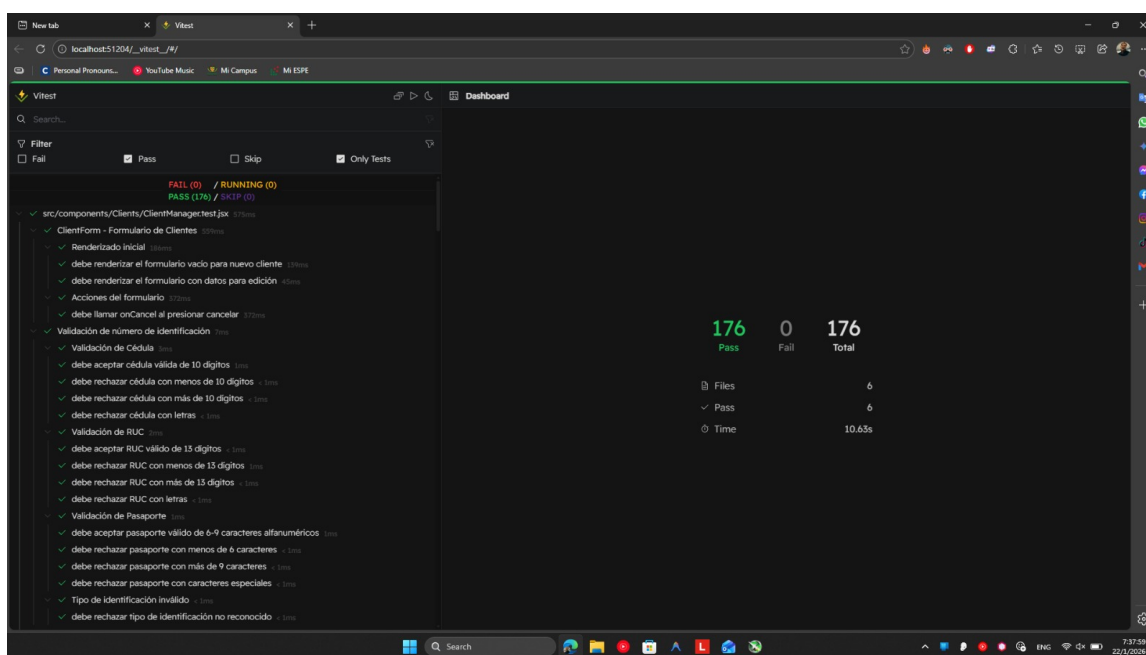
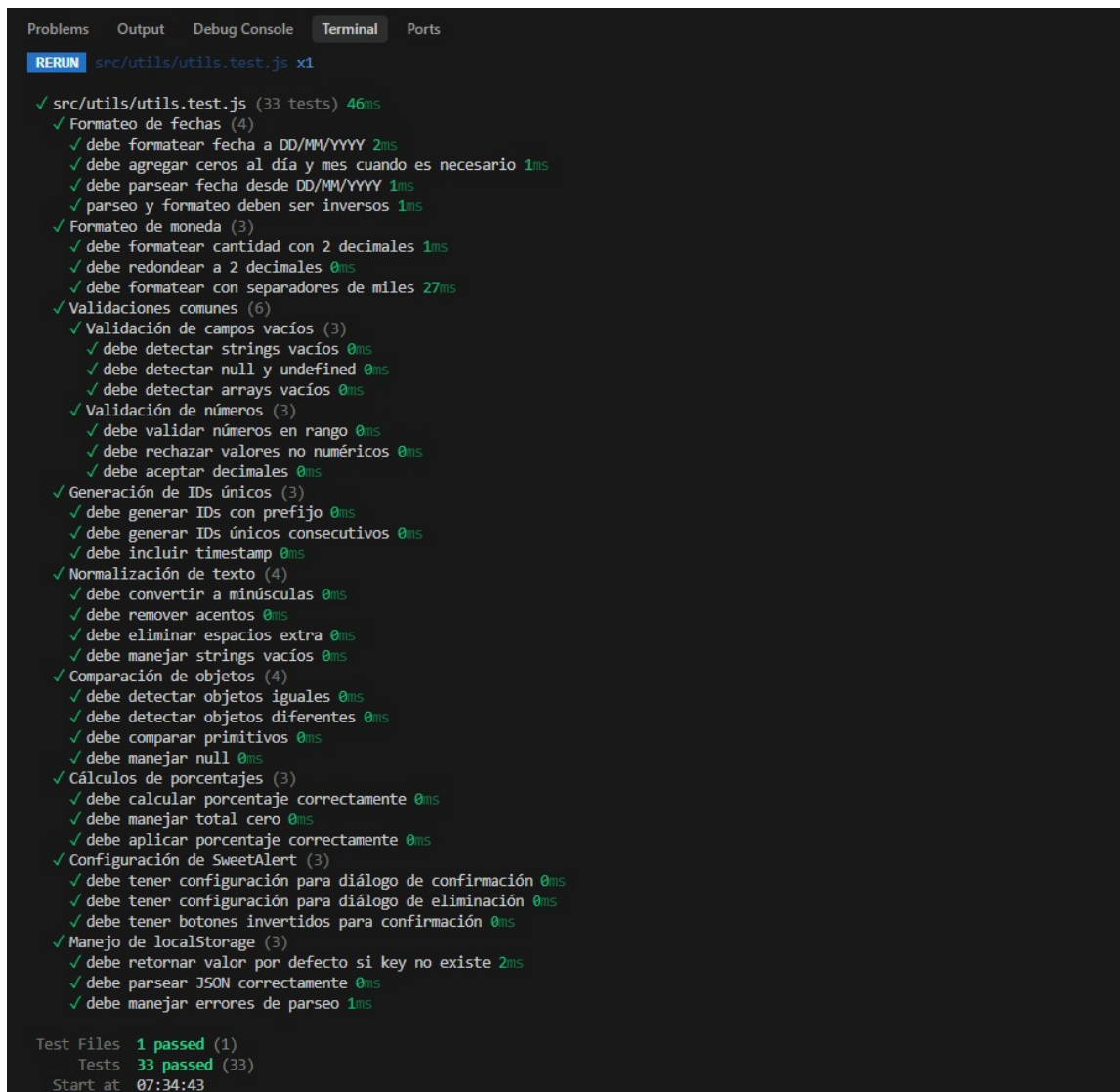


Figura 1: Vista general de Vitest UI — 176 pruebas pasaron sin errores



```
Problems Output Debug Console Terminal Ports
RERUN src/utils/utils.test.js x1

✓ src/utils/utils.test.js (33 tests) 46ms
✓ Formateo de fechas (4)
  ✓ debe formatear fecha a DD/MM/YYYY 2ms
  ✓ debe agregar ceros al día y mes cuando es necesario 1ms
  ✓ debe parsear fecha desde DD/MM/YYYY 1ms
  ✓ parseo y formateo deben ser inversos 1ms
✓ Formateo de moneda (3)
  ✓ debe formatear cantidad con 2 decimales 1ms
  ✓ debe redondear a 2 decimales 0ms
  ✓ debe formatear con separadores de miles 27ms
✓ Validaciones comunes (6)
  ✓ Validación de campos vacíos (3)
    ✓ debe detectar strings vacíos 0ms
    ✓ debe detectar null y undefined 0ms
    ✓ debe detectar arrays vacíos 0ms
  ✓ Validación de números (3)
    ✓ debe validar números en rango 0ms
    ✓ debe rechazar valores no numéricos 0ms
    ✓ debe aceptar decimales 0ms
✓ Generación de IDs únicos (3)
  ✓ debe generar IDs con prefijo 0ms
  ✓ debe generar IDs únicos consecutivos 0ms
  ✓ debe incluir timestamp 0ms
✓ Normalización de texto (4)
  ✓ debe convertir a minúsculas 0ms
  ✓ debe remover acentos 0ms
  ✓ debe eliminar espacios extra 0ms
  ✓ debe manejar strings vacíos 0ms
✓ Comparación de objetos (4)
  ✓ debe detectar objetos iguales 0ms
  ✓ debe detectar objetos diferentes 0ms
  ✓ debe comparar primitivos 0ms
  ✓ debe manejar null 0ms
✓ Cálculos de porcentajes (3)
  ✓ debe calcular porcentaje correctamente 0ms
  ✓ debe manejar total cero 0ms
  ✓ debe aplicar porcentaje correctamente 0ms
✓ Configuración de SweetAlert (3)
  ✓ debe tener configuración para diálogo de confirmación 0ms
  ✓ debe tener configuración para diálogo de eliminación 0ms
  ✓ debe tener botones invertidos para confirmación 0ms
✓ Manejo de localStorage (3)
  ✓ debe retornar valor por defecto si key no existe 2ms
  ✓ debe parsear JSON correctamente 0ms
  ✓ debe manejar errores de parseo 1ms

Test Files 1 passed (1)
Tests 33 passed (33)
Start at 07:34:43
```

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

```

DEV v4.0.17 D:/Semestre VII/Análisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0
✓ src/data/seedData.test.js (20 tests) 26ms
stderr | src/utills/utills.test.js > Manejo de localStorage > debe manejar errores de parseo
Error: reading key from localStorage: SyntaxError: Unexpected token 'i', "invalid json" is not valid JSON
    at JSON.parse (<anonymous>)
    at safeGetFromStorage (D:/Semestre VII/Análisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0/src/utills/utills.test.js:302:26)
    at D:/Semestre VII/Análisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0/src/utills/utills.test.js:340:20
    at file:///D:/Semestre VII/Análisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0/node_modules/@vitest/runner/dist/index.js:145:
11
    at file:///D:/Semestre VII/Análisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0/node_modules/@vitest/runner/dist/index.js:915:
26
    at file:///D:/Semestre VII/Análisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0/node_modules/@vitest/runner/dist/index.js:1243:
20
    at new Promise (<anonymous>)
    at runWithTimeout (file:///D:/Semestre VII/Análisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0/node_modules/@vitest/runner/di
st/index.js:1209:10)
    at file:///D:/Semestre VII/Análisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0/node_modules/@vitest/runner/dist/index.js:1653:
37
    at Traces.$ (file:///D:/Semestre VII/Análisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0/node_modules/vitest/dist/chunks/trac
es.CommonQaMT.js:142:27)

✓ src/utills/utills.test.js (33 tests) 69ms
✓ src/components/CustomMix/CustomMixDesigner.test.jsx (32 tests) 29ms
✓ src/components/Orders/OrderManager.test.jsx (35 tests) 31ms
✓ src/components/Clients/ClientManager.test.jsx (34 tests) 44ms
✓ src/components/Products/ProductManager.test.jsx (22 tests) 1699ms

Test Files 6 passed (6)
Tests 176 passed (176)
Start at 07:32:18
Duration 5.58s (transform 972ms, setup 3.55s, import 2.73s, tests 2.30s, environment 12.25s)

PASS Waiting for file changes...
press h to show help, press q to quit

```

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

```

PS D:\Semestre VII\Análisis y Diseño\27837_G6_ADS\Biblioteca de Trabajo\4. CODIGO\KairosMix_V3.0> npx vitest run src/data/seedData.test.js
RUN v4.0.17 D:/Semestre VII/Análisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0
✓ src/data/seedData.test.js (20 tests) 23ms
✓ Datos de ejemplo - Productos (8)
  ✓ debe tener productos definidos 3ms
  ✓ cada producto debe tener campos requeridos 3ms
  ✓ códigos de producto deben ser únicos 0ms
  ✓ IDs de producto deben ser únicos 1ms
  ✓ precios deben ser números positivos 2ms
  ✓ stock inicial debe ser número entero positivo 1ms
  ✓ precio mayorista debe ser menor que minorista 0ms
  ✓ códigos de producto deben seguir formato correcto (Letra + 2 dígitos) 1ms
✓ Datos de ejemplo - Clientes (6)
  ✓ debe tener clientes definidos 1ms
  ✓ cada cliente debe tener campos requeridos 1ms
  ✓ IDs de cliente deben ser únicos 0ms
  ✓ emails deben tener formato válido 0ms
  ✓ tipos de documento deben ser válidos 2ms
  ✓ debe haber al menos un cliente de cada tipo de documento 1ms
✓ Formato de fechas en seed data (3)
  ✓ debe formatear fecha correctamente 0ms
  ✓ debe agregar ceros a día de un dígito 0ms
  ✓ debe agregar ceros a mes de un dígito 0ms
✓ Consistencia de datos (3)
  ✓ productos y clientes no deben compartir IDs 0ms
  ✓ nombres de productos no deben estar vacíos 0ms
  ✓ nombres de clientes no deben estar vacíos 0ms

Test Files 1 passed (1)
Tests 20 passed (20)
Start at 07:35:51
Duration 2.95s (transform 97ms, setup 513ms, import 44ms, tests 23ms, environment 1.97s)

PS D:\Semestre VII\Análisis y Diseño\27837_G6_ADS\Biblioteca de Trabajo\4. CODIGO\KairosMix_V3.0> 

```

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