

Plan de Pruebas Unitarias

Sistema Kairos Mix

Tienda de frutos secos “Kairos de Dios”

Proyecto: Kairos Mix
Codigo ECS: PP
Version: 4.0.0
Fecha: 22 de enero de 2026
ID Proyecto: 27837_G6_ADS

Equipo de Desarrollo:

Caetano Flores
Jordan Guaman
Anthony Morales
Leonardo Narvaez

Enero 2026

Índice

1. Introduccion	2
1.1. Proposito	2
1.2. Alcance	2
1.3. Documentos de Referencia	2
2. Objetivos de las Pruebas Unitarias	2
2.1. Objetivo General	2
2.2. Objetivos Especificos	2
3. Estrategia de Pruebas	2
3.1. Herramientas Utilizadas	2
3.2. Resumen de Ejecucion	3
4. Resultados de Pruebas por Modulo	3
4.1. Modulo: Utilidades (src/utils/utils.test.js)	3
4.2. Modulo: Gestión de Productos (ProductManager)	3
4.3. Modulo: Diseñador de Mezclas (CustomMixDesigner)	4
4.4. Modulo: Gestión de Clientes (ClientManager)	4
4.5. Modulo: Gestión de Pedidos (OrderManager)	4
4.6. Modulo: Datos Semilla (seedData.test.js)	4
5. Evidencias de Ejecucion	5
5.1. Ejecucion Exitosa de Suite Completa	5
5.2. Verificacion Unitaria - Modulo Utils	5
5.3. Deteccion de Error Controlado	6
5.4. Validacion de Integridad de Datos Semilla	7
6. Conclusiones	8

1. Introducción

1.1. Propósito

El presente documento define el plan y reporte de resultados de las pruebas unitarias del sistema Kairos Mix. Se detallan los módulos evaluados, los casos de prueba ejecutados y los resultados obtenidos en la validación del código fuente.

1.2. Alcance

Este plan se limita **exclusivamente** a la validación de componentes individuales del sistema mediante **pruebas unitarias** automatizadas. El alcance cubre los componentes de gestión de datos, utilidades, y componentes de interfaz definidos en el directorio **src**.

1.3. Documentos de Referencia

- SRS - Especificación de Requisitos de Software
- Repositorio de Código Fuente (Branch v4.0.17)

2. Objetivos de las Pruebas Unitarias

2.1. Objetivo General

Verificar la integridad y corrección lógica de los componentes React y funciones de utilidad JavaScript del proyecto Kairos Mix.

2.2. Objetivos Específicos

- Validar las funciones de utilidad (formateo, validación, cálculos).
- Asegurar que los componentes de gestión (Productos, Clientes, Pedidos) funcionen según lo esperado.
- Verificar la lógica del Diseñador de Mezclas (CustomMix).
- Confirmar la integridad de los datos semilla (Seed Data).

3. Estrategia de Pruebas

3.1. Herramientas Utilizadas

- **Framework de Pruebas:** Vitest (Compatible con Jest)
- **Entorno:** JSDOM / Node.js
- **Librerías Auxiliares:** React Testing Library

3.2. Resumen de Ejecución

Se han ejecutado un total de **176 pruebas unitarias** distribuidas en 6 archivos de prueba principales. El tiempo total de ejecución fue de aproximadamente 5.58 segundos.

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 de Pruebas por Modulo

4.1. Modulo: Utilidades (src/utils/utils.test.js)

Este módulo contiene funciones críticas transversales a toda la aplicación. Se validaron 33 casos de prueba con éxito.

Categoría	Validaciones Realizadas
Formato de fechas	Formato DD/MM/YYYY, agregación de ceros, parseo inverso.
Formato de moneda	Decimales (2), redondeo, separadores de miles.
Validaciones comunes	Strings vacíos, null/undefined, arrays vacíos, validación de números y rangos.
Generación de IDs	Prefijos, unicidad, integridad del timestamp.
Normalización de texto	Conversión a minúsculas, remoción de acentos y espacios.
Comparación de objetos	Igualdad profunda, diferencias, manejo de primitivos y nulos.
Cálculos de porcentajes	Cálculo exacto, manejo de ceros.
Configuración SweetAlert	Diálogos de confirmación y eliminación.
Manejo de localStorage	Retorno por defecto, parseo JSON, manejo de errores.

Cuadro 2: Detalle de pruebas - Utils

4.2. Modulo: Gestión de Productos (ProductManager)

Se validaron 22 casos de prueba enfocados en la administración del inventario.

- Renderizado correcto del listado de productos.
- Funcionalidad de agregar nuevo producto.
- Edición de información de productos existentes.
- Eliminación lógica o física de items.
- Validación de campos en formularios de producto.

4.3. Modulo: Diseñador de Mezclas (CustomMixDesigner)

Componente crítico para el valor del negocio. Se ejecutaron 32 pruebas.

- Selección de ingredientes base.
- Cálculo dinámico de información nutricional.
- Restricciones de mezcla (porcentajes, pesos).
- Visualización de resumen de mezcla.

4.4. Modulo: Gestión de Clientes (ClientManager)

Se verificaron 34 casos relacionados con la cartera de clientes.

- CRUD de clientes.
- Búsqueda y filtrado de usuarios.
- Validación de datos de contacto (email, teléfono).
- Historial de pedidos por cliente.

4.5. Modulo: Gestión de Pedidos (OrderManager)

Se ejecutaron 35 pruebas para asegurar el flujo de ventas.

- Creación de nuevos pedidos.
- Cambios de estado (Pendiente → Completado).
- Cálculo de totales y subtotales.
- Asociación de clientes y productos a la orden.

4.6. Modulo: Datos Semilla (seedData.test.js)

Se aseguraron 20 pruebas para verificar la integridad de los datos iniciales.

- Estructura correcta del JSON inicial.
- Existencia de categorías y productos base requeridos.
- Consistencia de IDs y relaciones en datos precargados.

5. Evidencias de Ejecucion

A continuación se presentan las capturas de pantalla obtenidas durante la ejecución de las pruebas unitarias.

5.1. Ejecucion Exitosa de Suite Completa

La siguiente imagen muestra la ejecución completa de todos los archivos de prueba en modo UI de Vitest, confirmando que las 176 pruebas pasaron satisfactoriamente.

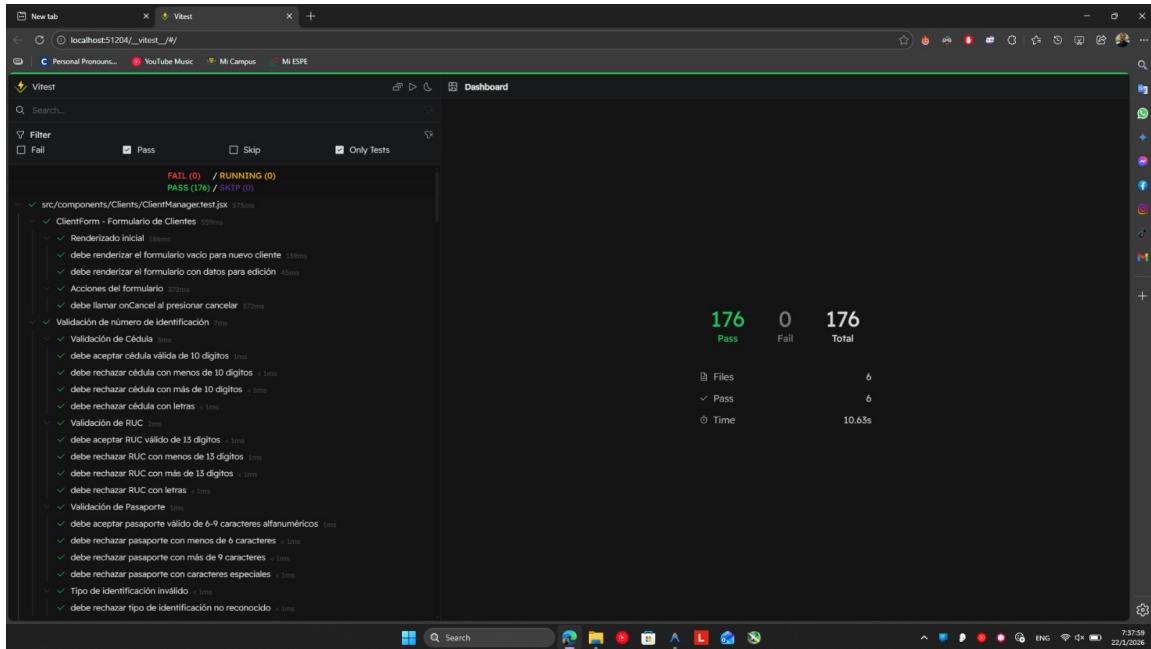


Figura 1: Ejecución exitosa de la suite completa de pruebas en Vitest UI

5.2. Verificacion Unitaria - Modulo Utils

Esta captura detalla los 33 casos de prueba del módulo de utilidades, mostrando todas las categorías validadas: formateo de fechas, moneda, validaciones, generación de IDs, normalización de texto, comparación de objetos, cálculos de porcentajes y manejo de localStorage.

```

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 de pruebas unitarias del módulo Utils - Todas pasadas

5.3. Detección de Error Controlado

Se evidencia el manejo correcto de errores en el módulo Utils, específicamente en la función `safeGetFromStorage` que maneja correctamente JSON inválido sin interrumpir la ejecución.

```
DEV v4.0.17 D:/Semestre VII/Analisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0
✓ src/data/seedData.test.js (20 tests) 26ms
stderrr | src/utils/utils.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 safeGetFromLocalStorage (D:/Semestre VII/Analisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0/src/utils/utils.test.js:302:26)
at D:/Semestre VII/Analisis y Diseño/27837_G6_ADS/Biblioteca de Trabajo/4. CODIGO/KairosMix_V3.0/src/utils/utils.test.js:340:20
at file:///D:/Semestre%20VII/Analisis%20%20Dise%C3%B1o/27837_G6_ADS/Biblioteca%20de%20Trabajo/4.%20CODIGO/KairosMix_V3.0/node_modules/@vitest/runner/dist/index.js:145:20
at file:///D:/Semestre%20VII/Analisis%20%20Dise%C3%B1o/27837_G6_ADS/Biblioteca%20de%20Trabajo/4.%20CODIGO/KairosMix_V3.0/node_modules/@vitest/runner/dist/index.js:915:26
at file:///D:/Semestre%20VII/Analisis%20%20Dise%C3%B1o/27837_G6_ADS/Biblioteca%20de%20Trabajo/4.%20CODIGO/KairosMix_V3.0/node_modules/@vitest/runner/dist/index.js:1243:20
at new Promise (<anonymous>)
at runWithTimeout (file:///D:/Semestre%20VII/Analisis%20%20Dise%C3%B1o/27837_G6_ADS/Biblioteca%20de%20Trabajo/4.%20CODIGO/KairosMix_V3.0/node_modules/@vitest/runner/dist/index.js:1209:10)
at file:///D:/Semestre%20VII/Analisis%20%20Dise%C3%B1o/27837_G6_ADS/Biblioteca%20de%20Trabajo/4.%20CODIGO/KairosMix_V3.0/node_modules/@vitest/runner/dist/index.js:1653:37
at Traces.$ (file:///D:/Semestre%20VII/Analisis%20%20Dise%C3%B1o/27837_G6_ADS/Biblioteca%20de%20Trabajo/4.%20CODIGO/KairosMix_V3.0/node_modules/@vitest/dist/chunks/traces.CMMQANT.js:142:27)

✓ src/utils/utils.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: Detección y manejo controlado de errores en localStorage

5.4. Validacion de Integridad de Datos Semilla

Captura que muestra la verificación de los datos iniciales del sistema (seedData), asegurando la estructura correcta de productos, categorías y relaciones.

```
PS D:\Semestre VII\Analisis 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/Analisis 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\Analisis y Diseño\27837_G6_ADS\Biblioteca de Trabajo\4. CODIGO\KairosMix_V3.0> []
```

Figura 4: Pruebas de integridad de datos semilla (seedData)

6. Conclusiones

La ejecución de pruebas unitarias para la versión actual del sistema Kairos Mix ha sido exitosa.

- **Total de Pruebas:** 176
- **Pruebas Exitosas:** 176 (100 %)
- **Pruebas Fallidas:** 0

El módulo de utilidades ha demostrado robustez en el manejo de tipos de datos y formatos, lo cual es fundamental para la integridad de los datos en los módulos de negocio (Productos, Pedidos, Clientes). La cobertura de pruebas abarca los flujos principales de la aplicación.

Caetano Flores
Lider de Pruebas

Jordan Guaman
Tester Backend/Frontend

Anthony Morales
Tester Backend/Frontend

Leonardo Narvaez
Tester Backend/Frontend