Instituto Tecnológico de Costa Rica

Escuela de Ingeniería en Computación

IC6831 Aseguramiento De La Calidad Del Software

Grupo 40

Proyecto - Entrega final

Profesora:

Ericka Solano Fernández

Equipo 1:

Valery Mishel Carvajal Oreamuno – 2022314299

Luis Felipe Calderon Perez – 2021048663

Sergio Chavarría Avilés – 2021077879

Kevin Yadir Calvo Rodriguez – 2023367224

Andrew Denilson Lopez Herrera – 2021062132

Enero, 2025

Índice

Índice de Figuras

1. Alcance del documento

Este documento tiene como objetivo comunicar los resultados del plan de pruebas aplicado al proyecto "Sistema de Gestión de Inventarios y Ventas". Este informe está estructurado para documentar de manera formal las actividades realizadas, los resultados obtenidos y las recomendaciones derivadas del proceso de evaluación. Este informe abarca los siguientes aspectos clave del proyecto y el proceso de evaluación:

1. **Evaluación de la aplicación:**
    - Se analizaron los módulos principales del sistema, incluyendo Clientes, Proveedores, Ventas, Inventarios y Datos Estadísticos.
    - La evaluación abarcó atributos como funcionalidad, usabilidad y mantenibilidad, priorizando escenarios críticos relacionados con la operación diaria del sistema.

2. **Propósito del plan:**
    - Validar el cumplimiento de diversos requerimientos del proyecto.
    - Identificar defectos y posibles áreas de mejora en el diseño, desarrollo y ejecución del sistema.
    - Proveer recomendaciones para la optimización futura del sistema.

3. **Cobertura del plan de pruebas:**
    - **Pruebas Estáticas:** Inspecciones de código para evaluar complejidad ciclomática y duplicación de código.
    - **Pruebas Dinámicas:**
        - Unitarias: Validación de correctitud y protección contra errores en funciones específicas.
        - Integración: Evaluación de la comunicación entre módulos del backend y el frontend.
        - Sistema: Verificación de la funcionalidad integral en escenarios de uso real.
        - Aceptación de Usuario: Pruebas automatizadas y manuales enfocadas en tareas clave dentro de cada módulo.

4. **Resultados esperados:**
    - Identificar defectos críticos que puedan impactar la operatividad del sistema.
    - Confirmar la conformidad de los resultados con los criterios de aceptación establecidos.

5. **Limitaciones:**
    - La evaluación se centra en los módulos mencionados y no abarca funcionalidades futuras o que estén fuera del alcance inicial del proyecto.
    - Las pruebas se realizaron en un entorno controlado que puede no reflejar todas las condiciones del entorno de producción.

2. Aplicación para evaluar

a. Contexto de la aplicación

La aplicación tiene como objetivo principal trabajar con la base de datos Wide World Importers, ofreciendo una plataforma web donde los usuarios puedan realizar consultas detalladas y personalizadas sobre diferentes módulos de la base de datos. La solución debe implementar procedimientos almacenados para garantizar que toda la lógica de búsqueda y transformación de datos ocurra del lado del servidor de la base de datos, dejando a la interfaz web únicamente la presentación de la información y el envío de parámetros.

La aplicación estará dividida en cuatro módulos principales: **clientes, proveedores, inventarios y ventas**. Cada uno permitirá realizar búsquedas a través de filtros acumulativos, presentando los resultados en tablas ordenadas alfabéticamente. Los usuarios tendrán la opción de seleccionar un elemento de la tabla para obtener una vista detallada de su información, que incluirá datos como contactos, métodos de entrega, ubicaciones (con mapas), entre otros.

Adicionalmente, la aplicación contará con un módulo de estadísticas, diseñado para generar reportes específicos como las compras más altas y bajas por proveedor, las ventas promedio por cliente, los productos más rentables por año, y los clientes y proveedores más destacados según órdenes y facturas.

b. Tecnologías en las que está implementada

Frontend: Se utilizó el framework Next.js, se utilizó este framework para aumentar el rendimiento de la aplicación. Este framework está basado en React.

Backend: Se utilizó Node.js para el backend, este framework se utilizó, debido a la estabilidad y gran cantidad de herramientas que permiten un mejor trabajo.

Base de datos: Se utilizó Microsoft SQL Server como requisito principal de la aplicación.

Servidor: Esta aplicación está montada sobre un servidor virtual de Azure. Se utilizó este servidor, debido a la estabilidad de estos servidores.

c. Delimitación de las funcionalidades de enfoque.

Consulta y Filtros de Clientes: Implementar una interfaz que permita buscar y filtrar clientes por diferentes criterios, como nombre, ubicación, historial de compras, entre otros.

Detalle de Cliente: Mostrar información completa y específica de un cliente seleccionado, incluyendo datos personales, historial de compras y preferencias.

Consulta y Filtros de Proveedores: Habilitar la búsqueda y filtrado de proveedores con base en criterios como nombre, ubicación o productos suministrados.

Detalle de Proveedor: Visualizar toda la información relevante sobre un proveedor, como contactos, productos suministrados y rendimiento.

Consulta y Filtros de Productos: Proporcionar una interfaz para buscar y filtrar productos según criterios como nombre, categoría, precio o inventario disponible.

Detalle de Producto: Presentar información detallada sobre un producto específico, incluyendo descripción, precio, cantidad en inventario y proveedores relacionados.

Consulta y Filtros de Ventas: Implementar funcionalidades para buscar y filtrar ventas por fechas, clientes, productos vendidos o métodos de pago.

Detalle de Venta: Desplegar toda la información relacionada con una venta específica, como cliente, productos adquiridos, monto total y fecha de la transacción.

Análisis de Compras por Proveedor y Categoría: Generar reportes que detallen las compras realizadas, clasificadas por proveedor y categoría, para evaluar tendencias y costos.

Análisis de Ventas por Cliente y Categoría: Producir reportes que muestren las ventas organizadas por cliente y categoría de producto, para identificar patrones de consumo.

Top 5 de Productos más Rentables: Calcular y mostrar los cinco productos que generan mayores beneficios en términos de rentabilidad.

Top 5 de Clientes con Mayor Actividad: Identificar y listar los cinco clientes que realizan más compras o tienen mayor interacción con el sistema.

Top 5 de Proveedores con Mayor Actividad: Mostrar los cinco proveedores que suministran la mayor cantidad de productos o servicios.

Gestión de Inventarios: Implementar herramientas para monitorear y actualizar el inventario, con alertas para productos con stock bajo.

Comunicación Frontend-Backend mediante Servicios API: Garantizar que la comunicación entre la interfaz de usuario y el backend se realice a través de servicios API bien definidos y seguros.

3. Propósito del plan.

El propósito de este plan de pruebas es garantizar que el "Sistema de Gestión de Inventarios y Ventas" cumpla con los estándares de calidad definidos, tanto en sus requisitos funcionales como en los no funcionales, ofreciendo un sistema robusto, confiable y preparado para su implementación en un entorno productivo. Este documento establece un enfoque integral para identificar defectos, validar las funcionalidades principales y evaluar el rendimiento y la usabilidad del sistema en escenarios reales y simulados.

En particular, el plan de pruebas se diseñó para evaluar los módulos principales del sistema, como Clientes, Proveedores, Ventas e Inventarios, asegurando que estos operen de manera efectiva y cumplan con las expectativas de los usuarios finales. También se busca garantizar que los atributos no funcionales, como la seguridad, la mantenibilidad y el rendimiento, estén alineados con los requerimientos del proyecto.

En caso de que las pruebas realizadas muestren que el sistema cumple parcialmente con los requerimientos evaluados, este plan también tiene el propósito de identificar oportunidades de mejora, especialmente en áreas críticas como la protección contra errores del usuario, la complejidad del código y la capacidad del sistema para manejar escenarios de alta concurrencia.

Por lo tanto, este plan no solo sirve para verificar la calidad actual del sistema, sino también para proporcionar una base sólida que permita implementar mejoras futuras, orientadas a optimizar la experiencia del usuario, reducir la ocurrencia de errores y garantizar que el sistema sea escalable y mantenible a lo largo del tiempo.

4. Detalle del plan propuesto

a. Atributos del modelo definido

Atributo del modelo definido				
Categoría - Atributo	**Tipo de prueba**	**Instrumentos**	**Métricas**	**Criterios de aceptación**
Maintainability - Reusability	Estática de Recorrido	Herramienta SonarQube: https://www.sonarsource.com/products/sonarqube/	Complejidad ciclomática	M >= 20 : Inaceptable 10 <= M < 20 : Regular M < 10 : Aceptable
Maintainability - Modifiability	Estática de Inspección	Herramienta SonarQube: https://www.sonarsource.com/products/sonarqube/	Tasa de duplicación de código	Duplicación >= 5% : Inaceptable Duplicación < 5% : Aceptable
Functional Suitability - Functional Correctness	Dinámica - Prueba Unitaria	Herramienta Jest: https://jestjs.io/	Correctitud	Correctitud = 100% : Aceptable
Usability - User Error Protection	Manual - Prueba Unitaria	Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P4)	Tasa de Error del Sistema	Aceptación: Si la Tasa de Error del Sistema es menor o igual al 50%. Rechazo: Si la Tasa de Error del Sistema es mayor al 50%.
Functional Suitability - Functional Correctness	Dinámica - De Integración	Herramienta Jest: https://jestjs.io/	Correctitud	Correctitud = 1 : Aceptable
Compatibility - Interoperability	Dinámica - De Integración	Herramienta Jest: https://jestjs.io/	Intercambiabilidad de formatos de datos	X≥ 0.97
Reliability - Availability	Dinámica - De Sistema	Herramienta UptimeRobot: https://uptimerobot.com/	Tiempo de actividad (Uptime)	Uptime >= 99% : Aceptable
Performance Efficiency - Capacity	Dinámica - De Sistema	Herramienta JMeter: https://jmeter.apache.org/	Usuarios concurrentes	Solicitudes aceptadas >= 99%
Usability - Accesibility	Dinámica - De Aceptación de Usuario	Prueba utilizando Lighthouse	Porcentaje de conformidad con WCAG	Normas cumplidas >= 50%
Functional Suitability - Functional Correctness	Dinámica - De Aceptación de Usuario	Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P10)	Porcentaje de aceptación del usuario	Aceptación: Si la aceptación es mayor o igual al 50%. Rechazo: Si la aceptación es menor al 50%.

b. Diseño de las pruebas

Diseño de las pruebas		
Diseño	**Instrumentalización**	**Escenarios**
Maintainability - Reusability. *Se le hará una prueba automatizada a todo el código para encontrar qué porcentaje del código se está duplicando.*	Herramienta SonarQube: https://www.sonarsource.com/products/sonarqube/	Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P1)
Maintainability - Modifiability. *Tasa de duplicación de código*	Herramienta SonarQube: https://www.sonarsource.com/products/sonarqube/	Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P2)
Functional Suitability - Functional Correctness. *Se harán tres pruebas unitarias en el backend que se utilizarán los filtros para probar el funcionamiento. Se contarán la cantidad de elementos devueltos y se comparan contra la cantidad de elementos que se deberían devolver, si son diferentes, entonces la prueba falla.*	Herramienta Jest: https://jestjs.io/	Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P3)
Usability - User Error Protection. *Se evaluará la capacidad del sistema para prevenir errores del usuario al ingresar datos inválidos en los filtros del módulo de ventas. Esto incluye validaciones de campos vacíos, formatos incorrectos, rangos no válidos y entradas incompatibles.*	Prueba manual: Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P4)	Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P4)
Functional Suitability - Functional Correctness. *Se harán pruebas de verificación de respuestas entre sus componentes.*	Herramienta Jest: https://jestjs.io/	Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P5)
Compatibility - Interoperability. *Grado de Intercambiabilidad de formatos de datos. Se pondrá a prueba la validación de los formatos de entrada (Codificación de textos, números y decimales) y salida (Formato de datos de respuesta y su flexibilidad). Además, de probar la tolerancia a formatos inesperados.*	Herramienta Jest: https://jestjs.io/	Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P6)

Reliability - Availability. *Se pondrá a prueba la cantidad de tiempo que está disponible el sistema mediante el monitoreo constante de caídas del sistema.*	Herramienta UptimeRobot: https://uptimerobot.com/	Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P7)
Performance Efficiency - Capacity. *Se pondrá a prueba la capacidad de usuarios que pueden manejar el sistema de manera simultánea.*	Herramienta JMeter: https://jmeter.apache.org/	Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P8)
Usability - Accesibility. *Se utilizará una herramienta que evaluará el nivel de accesibilidad que tiene el sitio web.*	Prueba utilizando Lighthouse	Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P9)
Functional Suitability - Functional Correctness. *Se evaluarán la completitud, facilidad de aprender a usar, protección a errores del usuario y correctitud mediante pruebas manuales simuladas realizadas por 10 usuarios de prueba.*	Prueba manual: Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P10)	Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P10)

c. Instrumentos de las pruebas

Pruebas estáticas

1. ***Complejidad Ciclomática:***

Herramienta SonarQube: https://www.sonarsource.com/products/sonarqube/

Configuración de la herramienta

Para la instalación de la herramienta, se utilizó un equipo con Linux y Docker.
En consola, se ejecutaron los siguientes comandos:

sudo docker pull sonarqube

sudo docker run -d --name sonarqube-db -e POSTGRES_USER=sonar -e POSTGRES_PASSWORD=sonar -e POSTGRES_DB=sonarqube postgres:alpine

sudo docker run -d --name sonarqube -p 9000:9000 --link sonarqube-db:db -e SONAR_JDBC_URL=jdbc:postgresql://db:5432/sonarqube -e SONAR_JDBC_USERNAME=sonar -e SONAR_JDBC_PASSWORD=sonar sonarqube

Con esto último, se debería de visualizar la herramienta de SonarQube en el localhost:9000

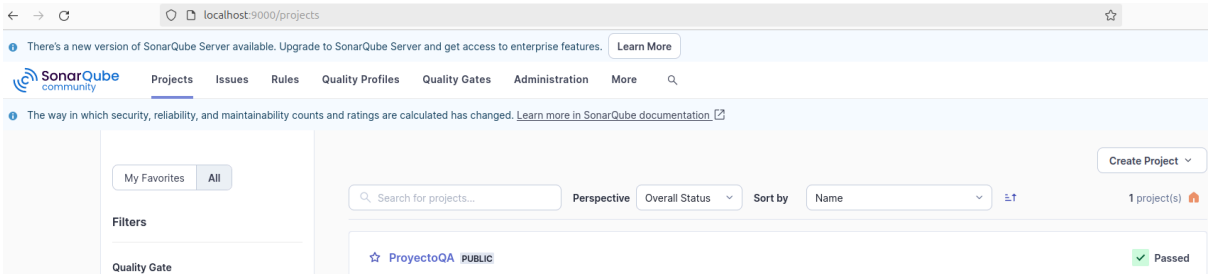

Figura 1
Visualización de la herramienta SonarQube en el puerto 9000

Luego, se procede a crear un proyecto, este se crea seleccionando la opción de "Create a local project"

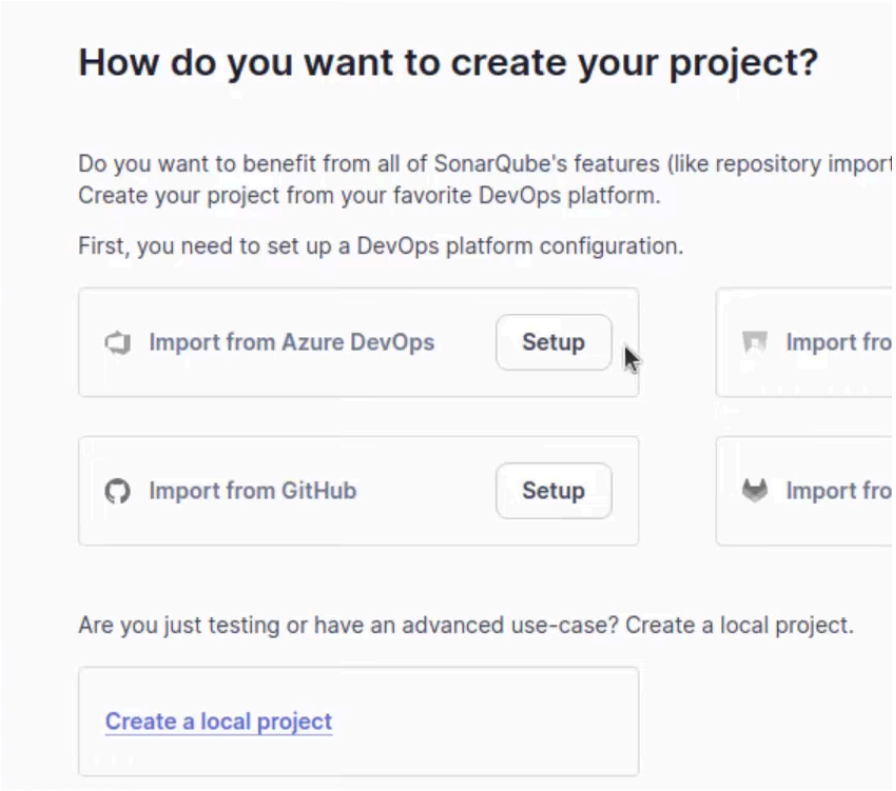

Figura 2
Selección de forma de crear un nuevo proyecto

Luego, se selecciona un nombre, en este caso, nuestro proyecto se llamará “ProyectoQA” Cuando nos piden seleccionar la base (Baseline) del proyecto, seleccionamos la opción de “Use the global setting”

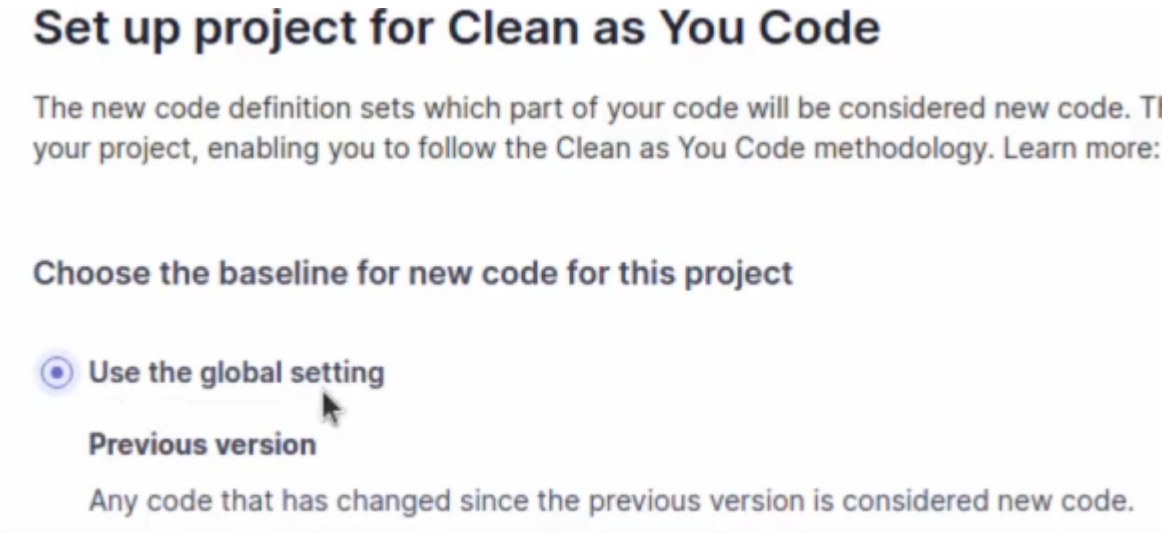

Figura 3
Selección de “Use the global setting” dentro de nuestro proyecto

Luego, cuando se nos pide cómo queremos analizar el proyecto, seleccionamos “Locally”

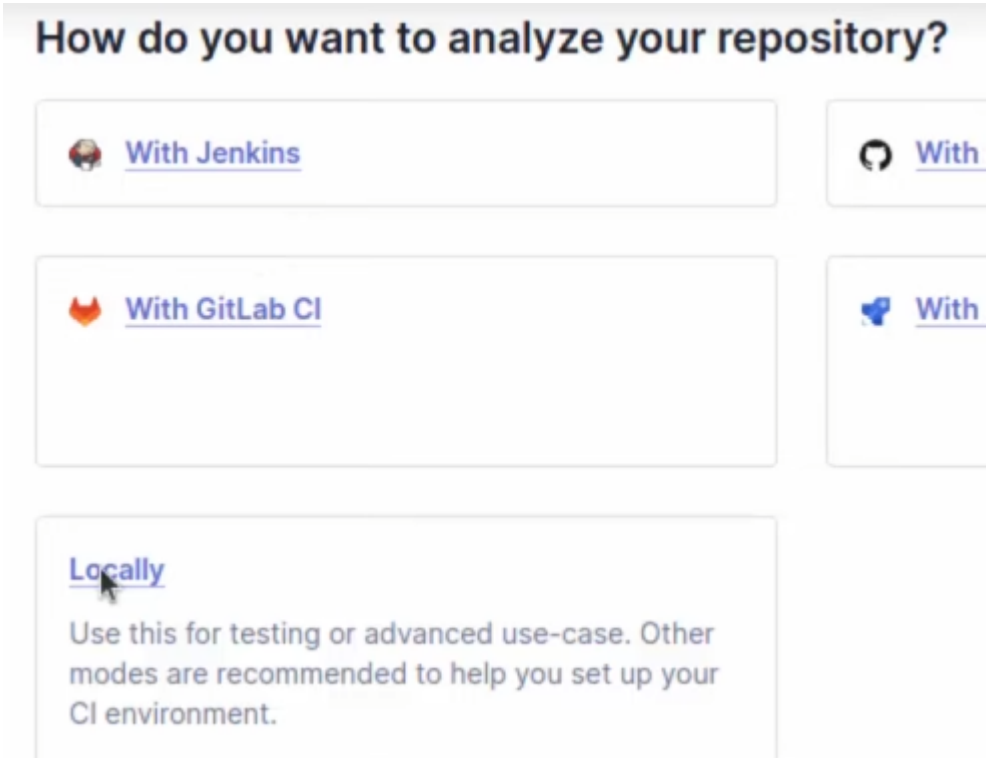

Figura 4
Selección de "Locally" como forma de analizar el repositorio

Finalmente tendremos que descargar los archivos binarios de el Sonar Scanner para poder analizar nuestro código. Hecho esto, debemos ejecutar el siguiente comando en consola en el directorio que queremos analizar:

```
/home/sergio/Downloads/sonar-scanner-cli-6.2.1.4610-linux-x64/sonar-scanner-6.2.1.4610-linux-x64/bin/sonar-scanner \
  -Dsonar.projectKey=ProyectoQA \
  -Dsonar.sources=. \
  -Dsonar.host.url=http://localhost:9000 \
  -Dsonar.token=sqp_95cc4238fa05dca48ea29a4c87e424ae03ea16cf
```

La primera línea debe contener el directorio hacia el archivo binario del Sonar Scanner, la segunda línea el nombre de nuestro proyecto, la tercera el lugar donde está nuestro SonarQube hosteado y la cuarta la token del proyecto que nos suministró SonarQube.

Ejecutando este comando, SonarQube nos generará un dashboard con toda la información de nuestro proyecto

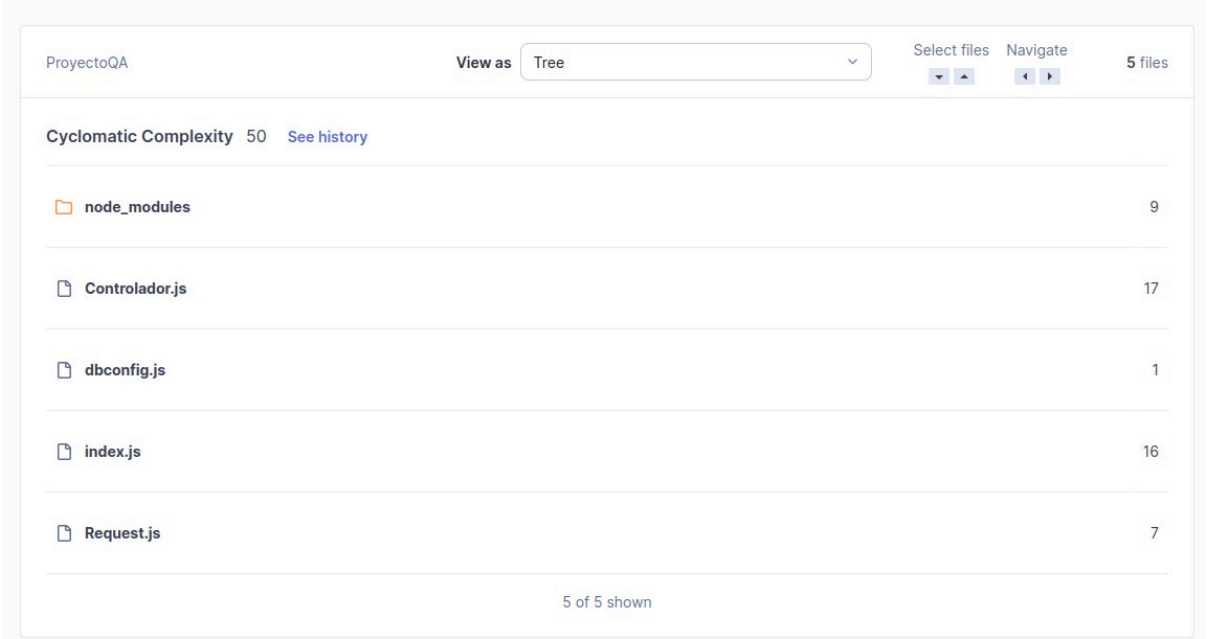

Figura 5
Resultados de la ejecución de las pruebas con SonarQube.

2. ***Tasa de duplicación de código:***

Herramienta SonarQube: https://www.sonarsource.com/products/sonarqube/

Configuración de la herramienta

El proceso de configuración es el mismo que para la prueba anterior.

Figura 6
Resultados de la ejecución de las pruebas con SonarQube.

Pruebas unitarias

1. ***Correctitud:***

Herramienta: Jest
Para esta prueba se decidió crear tres escenarios en los que se debía evaluar el nivel de correctitud en los resultados de la aplicación. Estos deben devolver una **X** datos en la prueba, si no devuelve la cantidad indicada, se considera fallida la prueba.

Escenario 1: Se desea filtrar el proveedor "A Datum Corporation" por el filtro del nombre.
Datos de entrada: Nombre: da

Escenario 2: Se desea filtrar el proveedor "Litware, Inc." por el filtro de categoría
Datos de entrada: Categoría: gi

Escenario 3: Se desea filtrar los proveedores A Datum Corporation, Fabrikam, Inc., Graphic Design Institute, Litware, Inc. y The Phone Company utilizando el filtro de nombre y el filtro de categoría.
Datos de entrada: Nombre: a , Categoría: g

Para cada una de estas pruebas se decidió programar una prueba utilizando Jest como herramienta para pruebas unitarias, se evaluó la clase Controlador, además se utilizó el método “VerEstadisticasProveedores(nombre, categoria)” para realizar las solicitudes al backend.

2. ***Tasa de error de usuarios:***

Herramienta: Prueba Manual
Para esta prueba, se identificaron 6 tareas a realizar en el sistema donde podrían surgir errores al ingresar datos erróneos, se llevaron a cabo dichas pruebas anotando los resultados.

Datos de la prueba:
Fórmula utilizada para medir la tasa de error del usuario:
Tasa de error del usuario = (Número de errores / Número total de intentos) x 100

Resultados obtenidos:
Número de errores: 4
Número total de intentos: 6
Tasa de error del usuario: 66,67%
Estado de la prueba: **RECHAZADO**

Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P4).

Tarea 1: Dejar todos los campos de filtro vacíos y hacer clic en "Solicitar Facturas".
Estado: Rechazado

Figura 7

Primer caso de prueba.

Figura 8

Primer caso de prueba.

Tarea 2: Ingresar un rango de fechas donde la fecha de inicio es posterior a la fecha de fin.
Estado: Rechazado

Figura 9
Segundo caso de prueba.

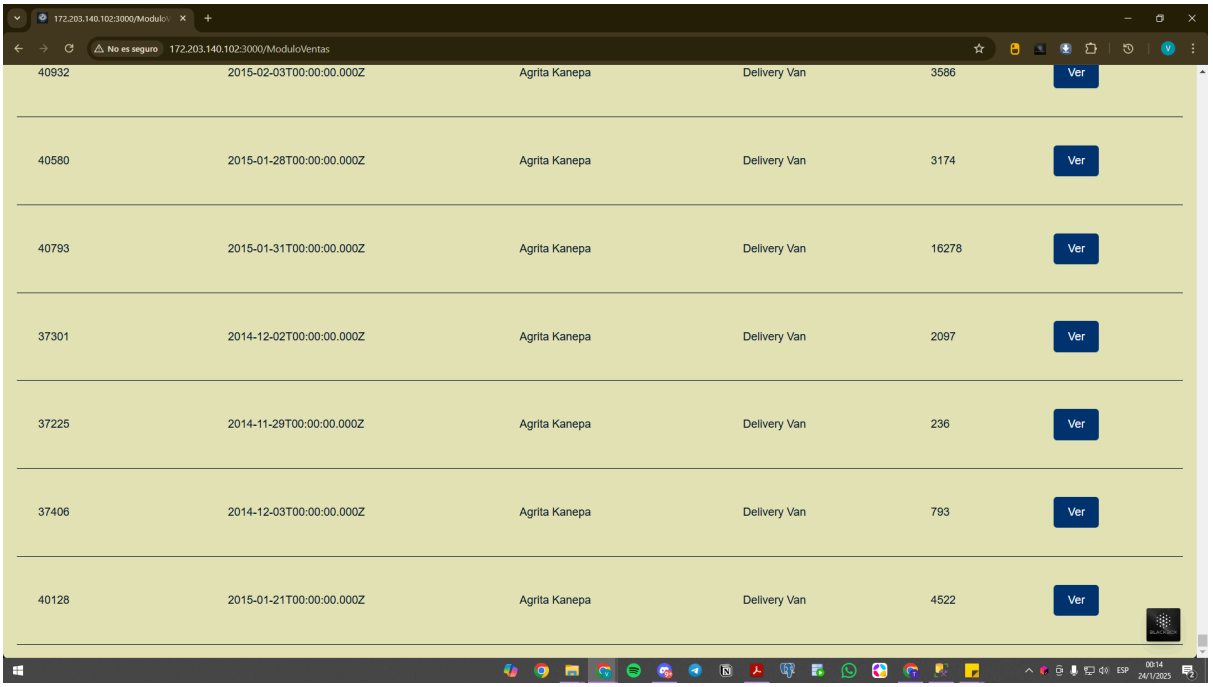

Figura 10
Segundo caso de prueba.

Tarea 3: Ingresar un rango de montos donde el monto mínimo es mayor que el monto máximo.
Estado: Rechazado

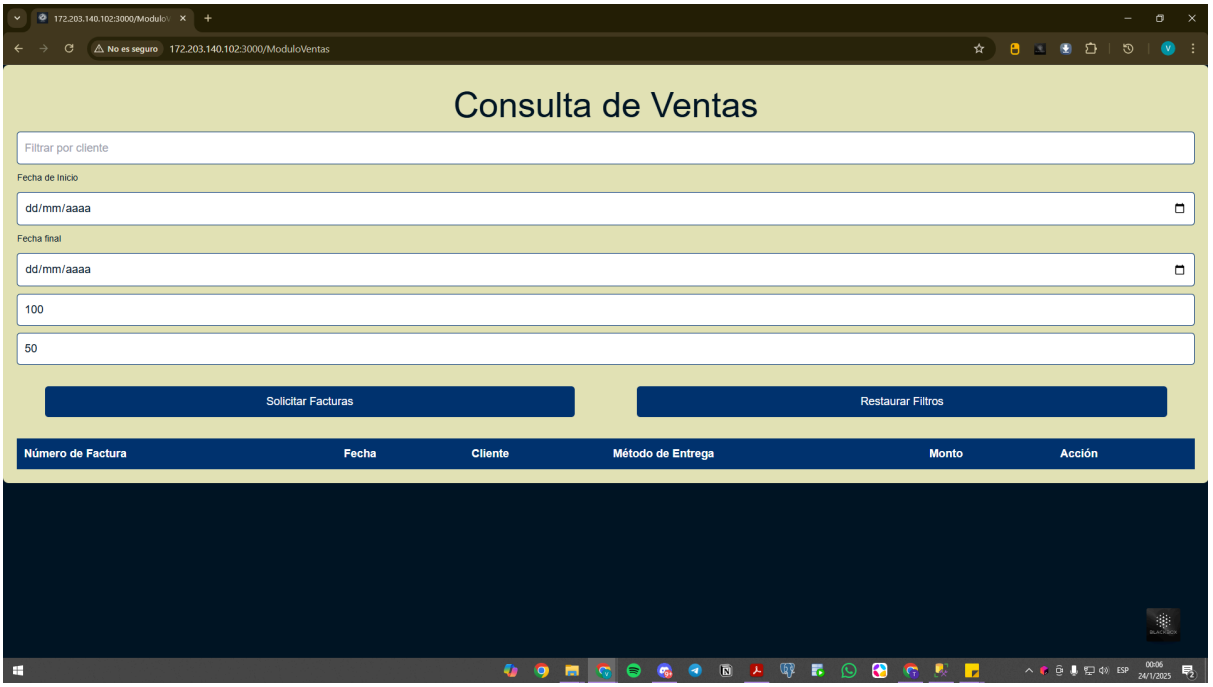

Figura 11
Tercer caso de prueba.

Tarea 4: Introducir caracteres no válidos (como texto) en los campos numéricos de monto.
Estado: Aprobado

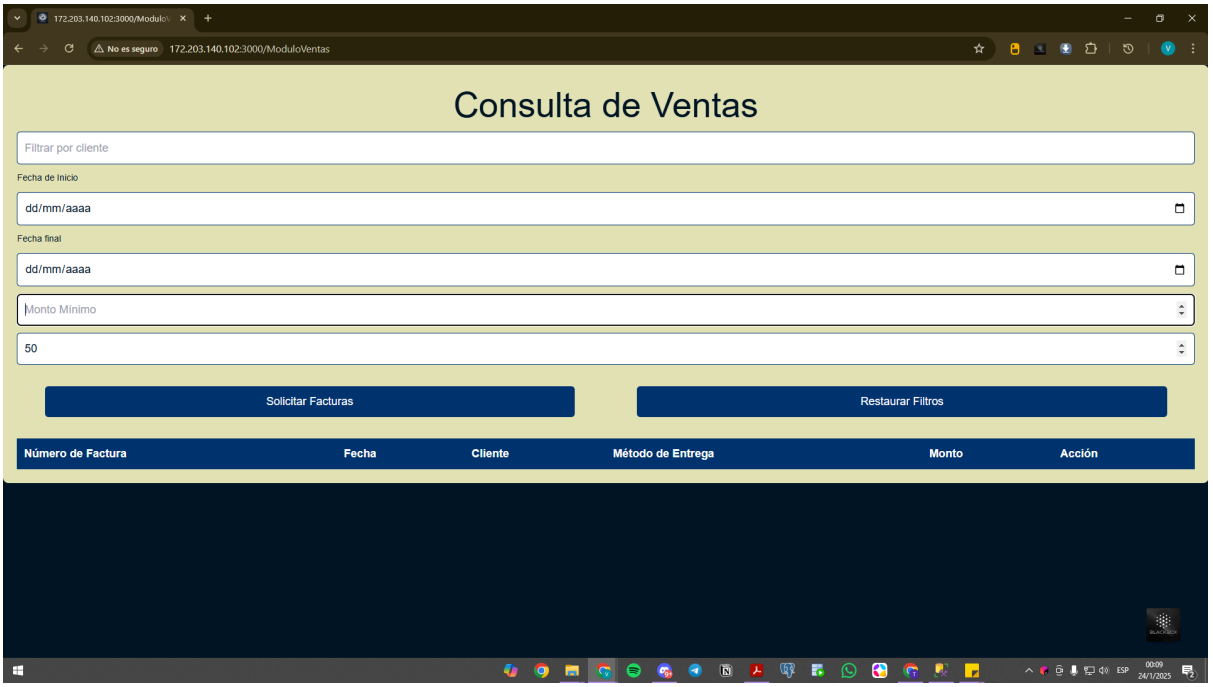

Figura 12
Cuarto caso de prueba.

Tarea 5: Introducir caracteres no válidos (como símbolos) en los campos numéricos de

monto.

Estado: Aprobado

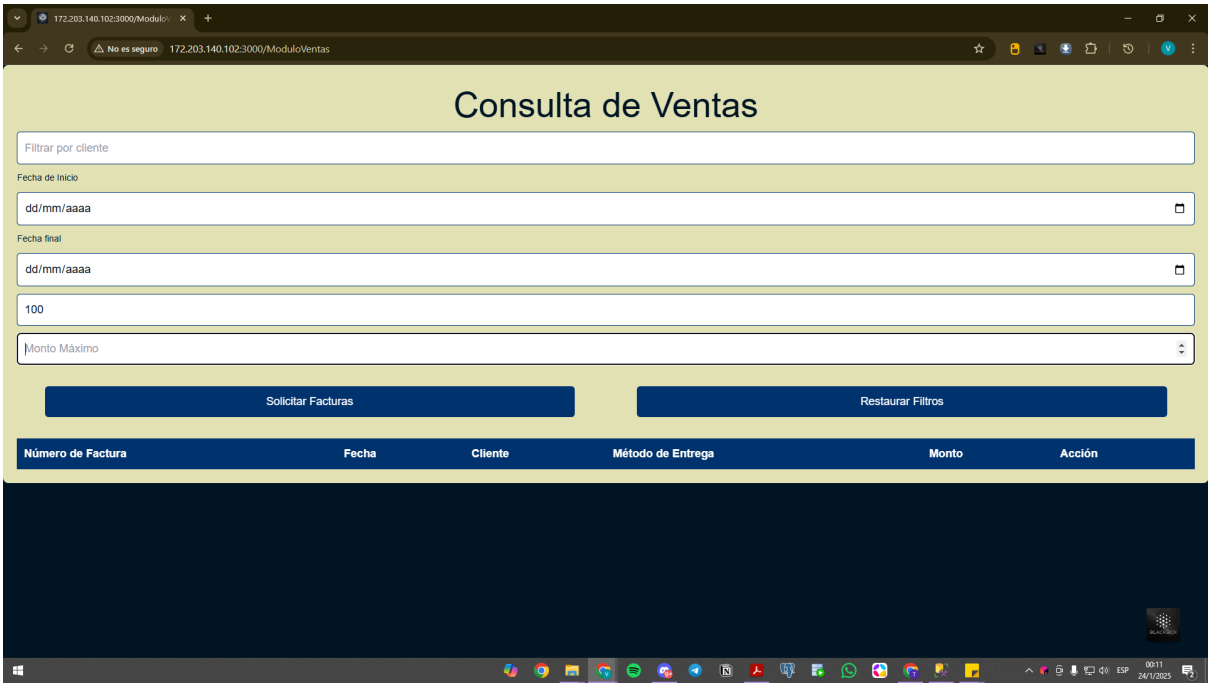

Figura 13

Quinto caso de prueba.

Tarea 6: Ingresar un número negativo en el rango de montos.

Estado: Rechazado

Figura 14

Sexto caso de prueba.

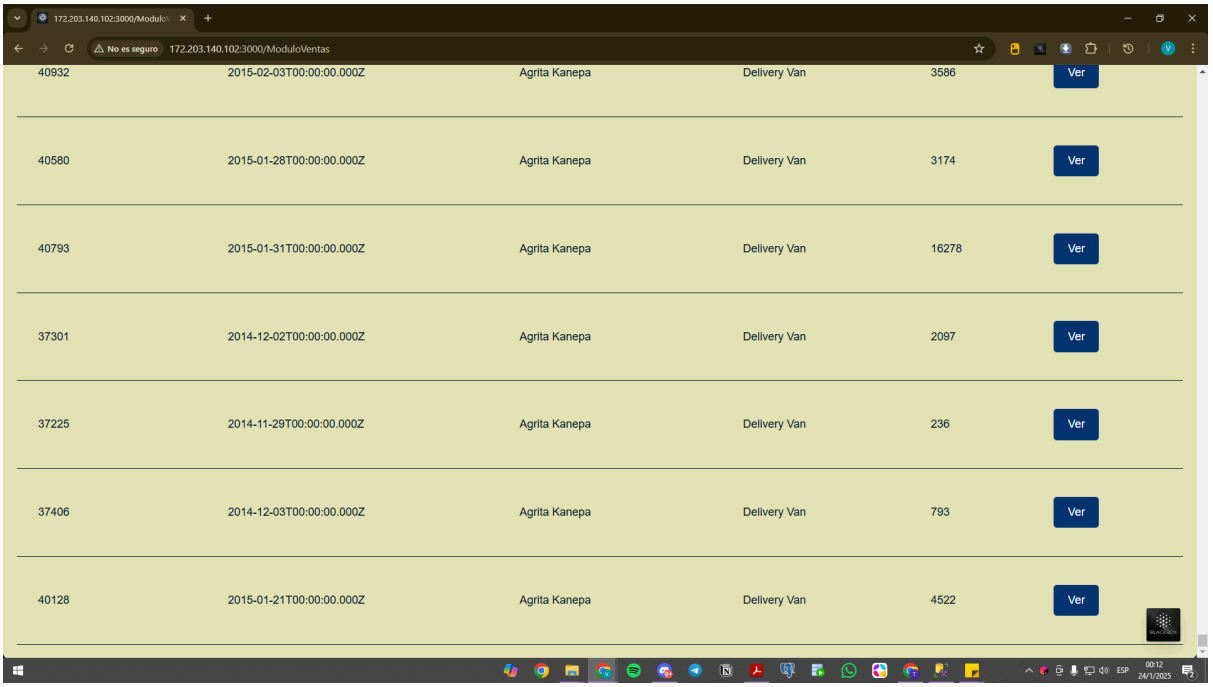

Figura 15
Sexto caso de prueba.

Pruebas de integración

Correctitud funcional y Grado de intercambiabilidad de formatos de datos:
Herramienta Jest: https://jestjs.io/

1. ***Correctitud funcional:***

Código fuente

```
const Controlador = require('../../Controlador');
const request = require('supertest');
const express = require('express');
const app = express();
const { sql } = require('../../dbconfig');

// Mock para dbconfig y mssql
jest.mock('../../dbconfig', () => ({
  sql: {
    Request: jest.fn(() => ({
      input: jest.fn(),
      execute: jest.fn().mockResolvedValue({
        recordset: [
          { Clientes: 'Cliente1', Calculo: 'Maximo', Valores: 1000 }
        ]
      })
    })),
    NVarChar: 'NVarCharMock',
  },
  connectToDatabase: jest.fn() // Mock de connectToDatabase
}));

test('VerEstadisticasProveedores devuelve datos correctamente', async () => {
  const controlador = new Controlador(sql);
  const result = await controlador.VerEstadisticasProveedores('Cliente1', 'Categoria1');

  expect(result).toEqual([{ Clientes: 'Cliente1', Calculo: 'Maximo', Valores: 1000 }]);
});

test('POST /MostrarEstadisticasProveedores devuelve estadísticas correctamente', async () => {
  const response = await request(app)
    .post('/MostrarEstadisticasProveedores')
    .send({ nombre: 'Cliente1', categoria: 'Categoria1' });

  expect(response.status).toBe(200);
  expect(response.body).toEqual([
    { Clientes: 'Cliente1', Calculo: 'Maximo', Valores: 1000 }
  ]);
});
```

Figura 16
Código fuente de las pruebas de integración en correctitud funcional.

Evidencia

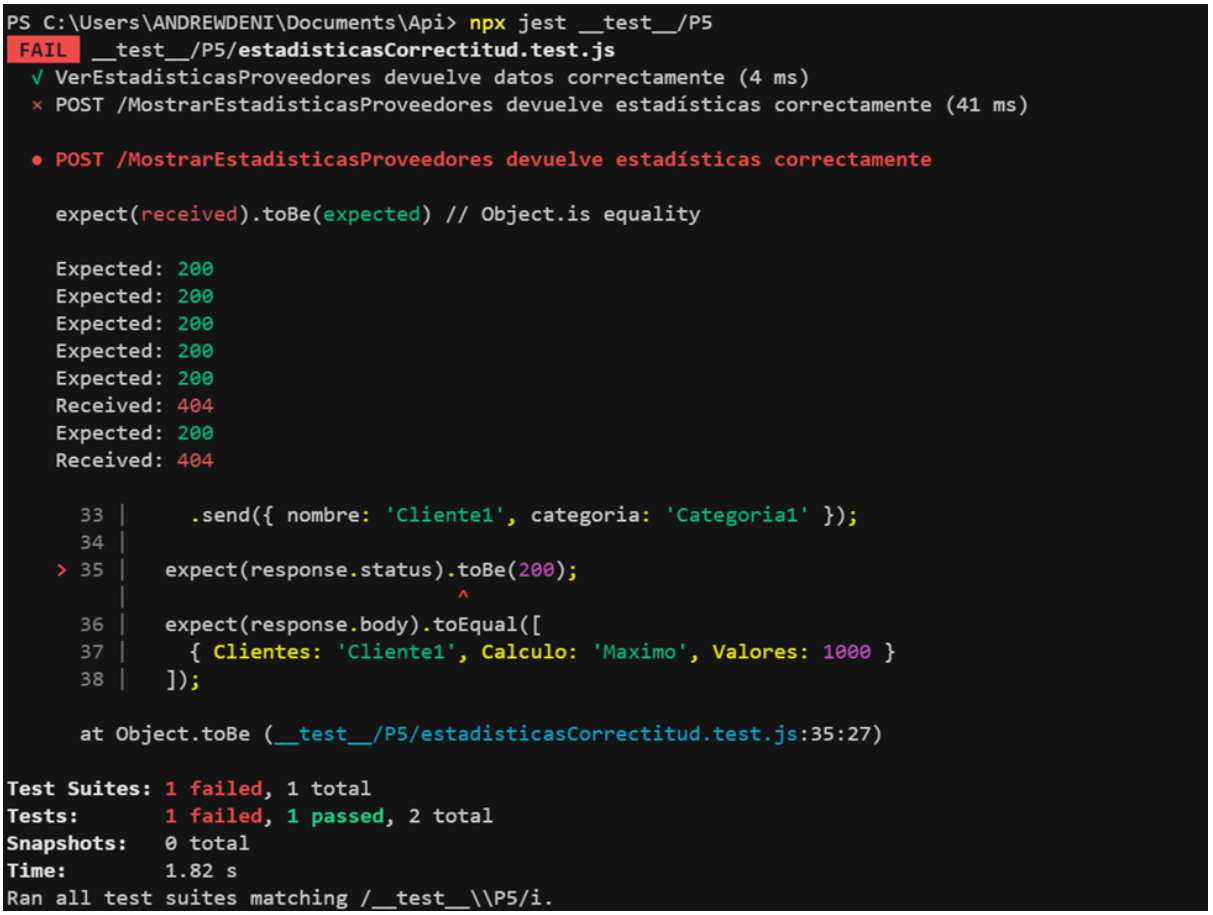

Figura 17

Resultados de pruebas de correctitud funciona con Jest.

2. Grado de intercambiabilidad de formatos de datos:

Código fuente

(módulo de ventas)

```
const request = require('supertest');
const app = require('../index');
const Controlador = require('../Controlador'); // Importamos el Controlador

// Hacemos mock del Controlador
jest.mock('../Controlador'); // Esto reemplaza el controlador original con un mock

describe('Pruebas de compatibilidad e interoperabilidad para /ConsultarVentas', ()
=> {

  // Aquí mockeamos el método ConsultarFacturas
  const mockConsultarFacturas = Controlador.prototype.ConsultarFacturas;

  beforeAll(() => {
    // Resetear mocks antes de cada test, por si acaso
    mockConsultarFacturas.mockReset();
  });
```

```
  it('Debería aceptar un filtro válido y devolver datos en el formato esperado',
async () => {
    // Configuramos la respuesta del mock
    mockConsultarFacturas.mockResolvedValueOnce([{
      ID: 1,
      FechaFactura: '2025-01-01',
      SitioWeb: 'example.com',
      NombreCliente: 'Adrian',
      MetodoEntrega: 'Envío a domicilio',
      MONTO: 3397
    }]);

    const response = await request(app)
      .post('/ConsultarVentas')
      .send({
        nombre: 'Adrian',
        fechaInicial: '2025-01-01',
        fechaFinal: '2025-01-30',
        montoInicial: '3396',
        montoFinal: '3398'
      });

    expect(response.status).toBe(200);
    expect(Array.isArray(response.body)).toBe(true);
    response.body.forEach((sale) => {
      expect(sale).toHaveProperty('ID');
      expect(sale).toHaveProperty('FechaFactura');
      expect(sale).toHaveProperty('SitioWeb');
      expect(sale).toHaveProperty('NombreCliente');
      expect(sale).toHaveProperty('MetodoEntrega');
      expect(sale).toHaveProperty('MONTO');
    });
  });

  it('Debería devolver un error 500 si el nombre es un entero y si el monto final
tiene letras', async () => {
    // Simulamos que el controlador da un error
    mockConsultarFacturas.mockRejectedValueOnce(new Error('Error en la base de
datos'));

    const response = await request(app)
      .post('/ConsultarVentas')
      .send({
        nombre: 8,
        fechaInicial: '2025-01-01',
        fechaFinal: '2025-01-30',
        montoInicial: '3396',
        montoFinal: 'adsv'
      });

    expect(response.status).toBe(500);
    expect(response.body).toHaveProperty('error');
  });

  it('Debería enviar un error porque no está el nombre', async () => {
    // Simulamos que el controlador da un error
    mockConsultarFacturas.mockRejectedValueOnce(new Error('Error en la base de
datos'));
```

```
    const response = await request(app)
      .post('/ConsultarVentas')
      .send({
        fechaInicial: '2025-01-01',
        fechaFinal: '2025-01-30',
        montoInicial: '3396',
        montoFinal: '3398'
      });

    expect(response.status).toBe(500);
    expect(response.body).toHaveProperty('error');
  });
});
```

(módulo de estadística proveedores)

```
const { sql, connectToDatabase } = require('../dbconfig');
const Controlador = require('../Controlador');

describe('VerEstadisticasProveedores', () => {
  let controlador;

  beforeAll(async () => {
    await connectToDatabase();
    controlador = new Controlador(sql);
  });

  afterEach(async () => {

  });

  it('Debería dar solamente un elemento para el primer conjunto de parámetros',
async () => {
    const result = await controlador.VerEstadisticasProveedores('da', '');
    expect(result).toHaveLength(1);
  });

  it('Debería dar solamente un elemento para el segundo conjunto de parámetros',
async () => {
    const result = await controlador.VerEstadisticasProveedores('', 'gi');
    expect(result).toHaveLength(1);
  });

  it('Debería dar cinco elementos para el tercer conjunto de parámetros', async ()
=> {
    const result = await controlador.VerEstadisticasProveedores('a', 'g');
    expect(result).toHaveLength(5);
  });
});
```

Evidencia

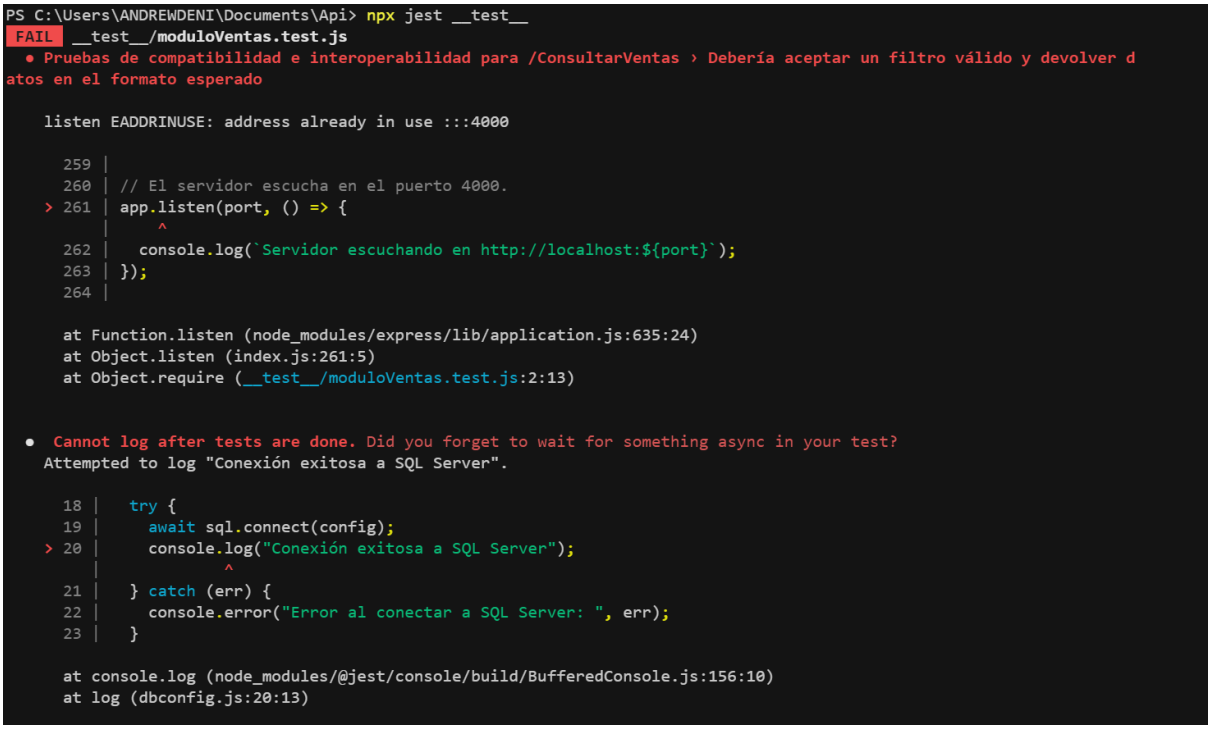

Figura 18

Resultados de pruebas intercambiabilidad de datos con Jest.

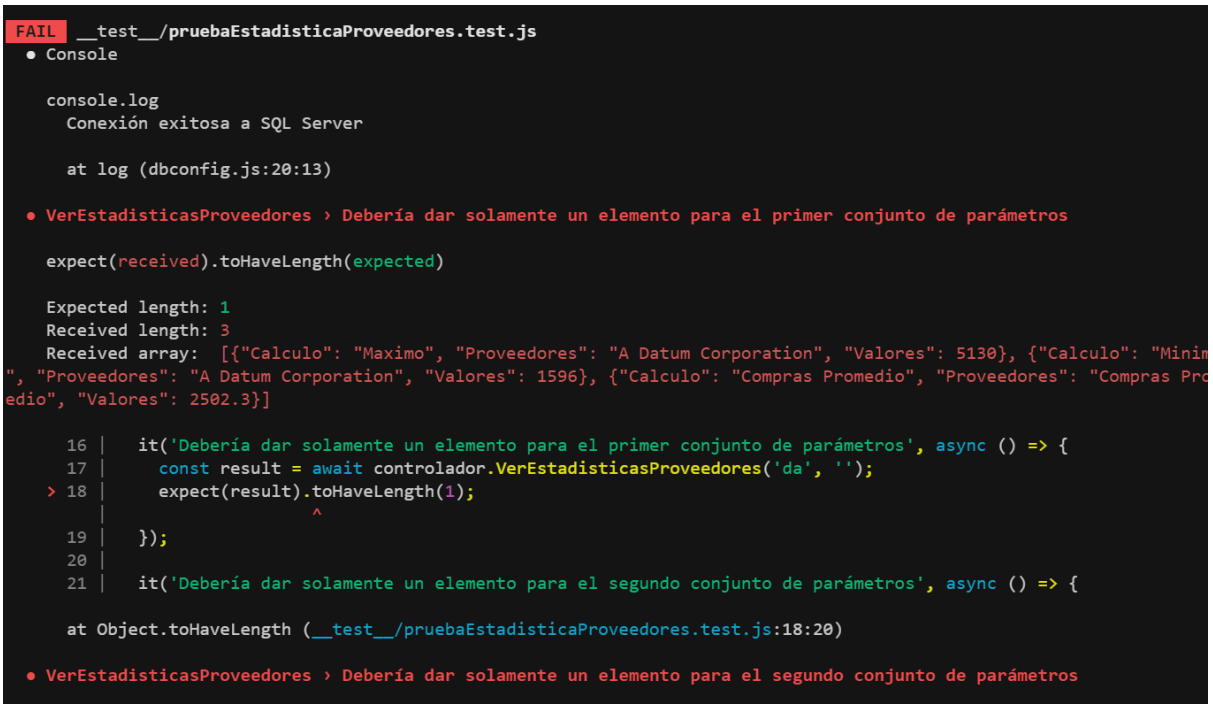

Figura 19

Resultados de pruebas intercambiabilidad de datos con Jest.

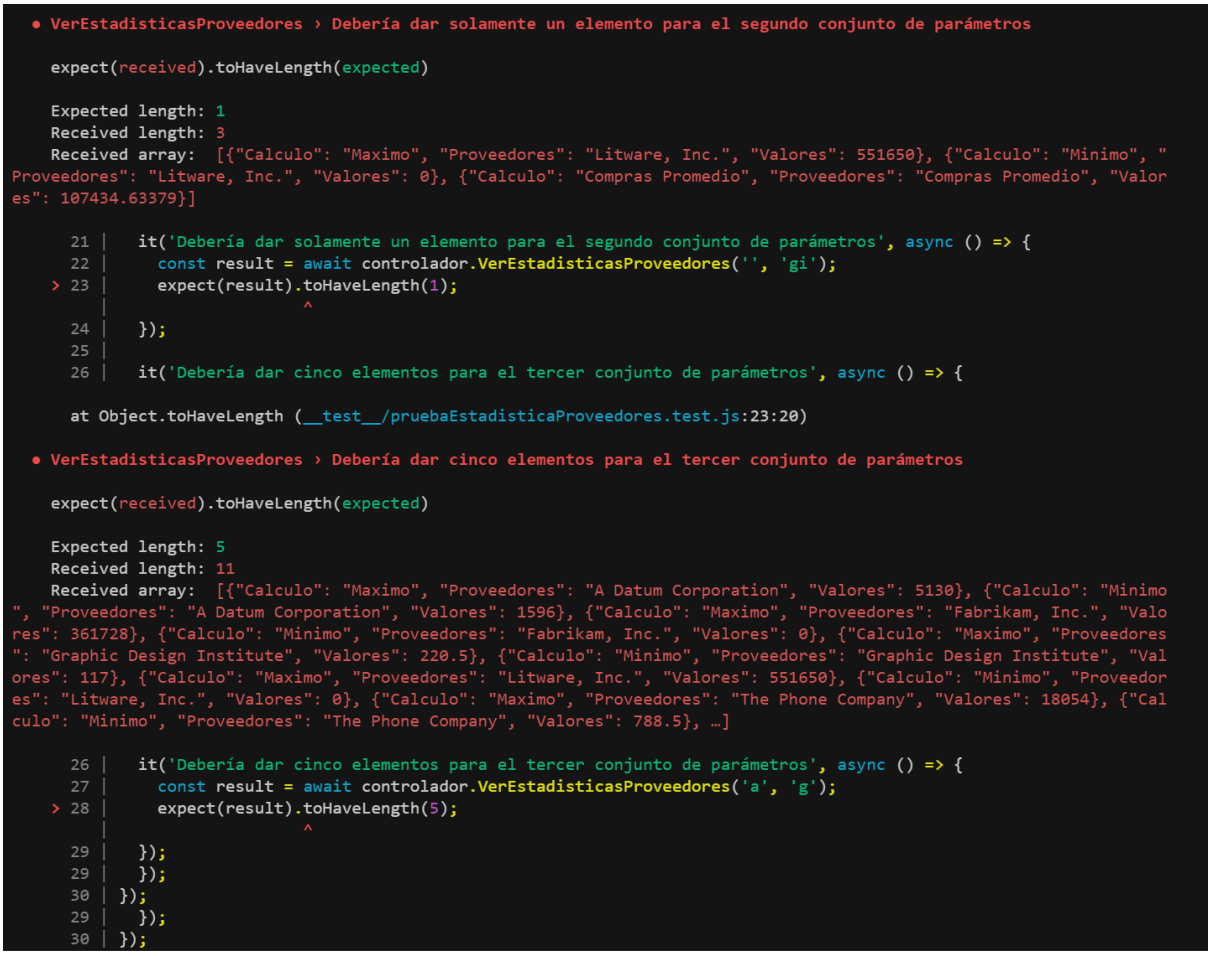

Figura 20

Resultados de pruebas intercambiabilidad de datos con Jest.

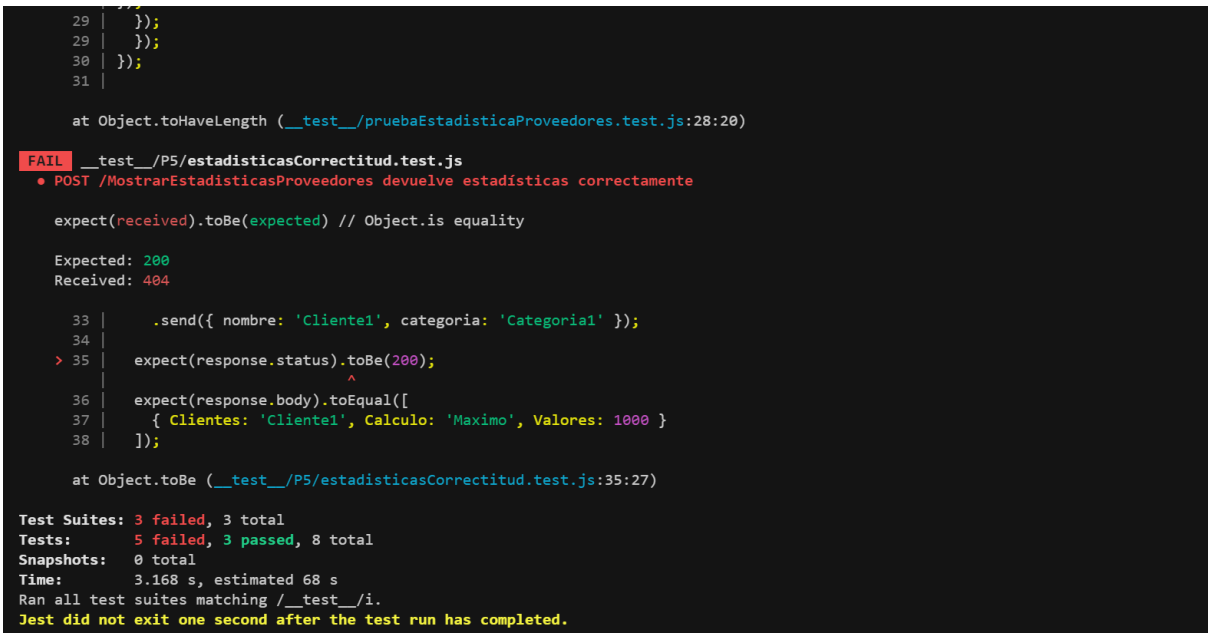

Figura 21

Resultados de pruebas intercambiabilidad de datos con Jest

Pruebas de sistema

1. ***Tiempo de actividad:***

Herramienta UptimeRobot: https://uptimerobot.com/

Configuración de la herramienta

Para la utilización de esta herramienta se debe crear una cuenta con un correo electrónico. Hecho esto, solamente se debe pasar el enlace a la página para que ésta comience un monitoreo continuo.

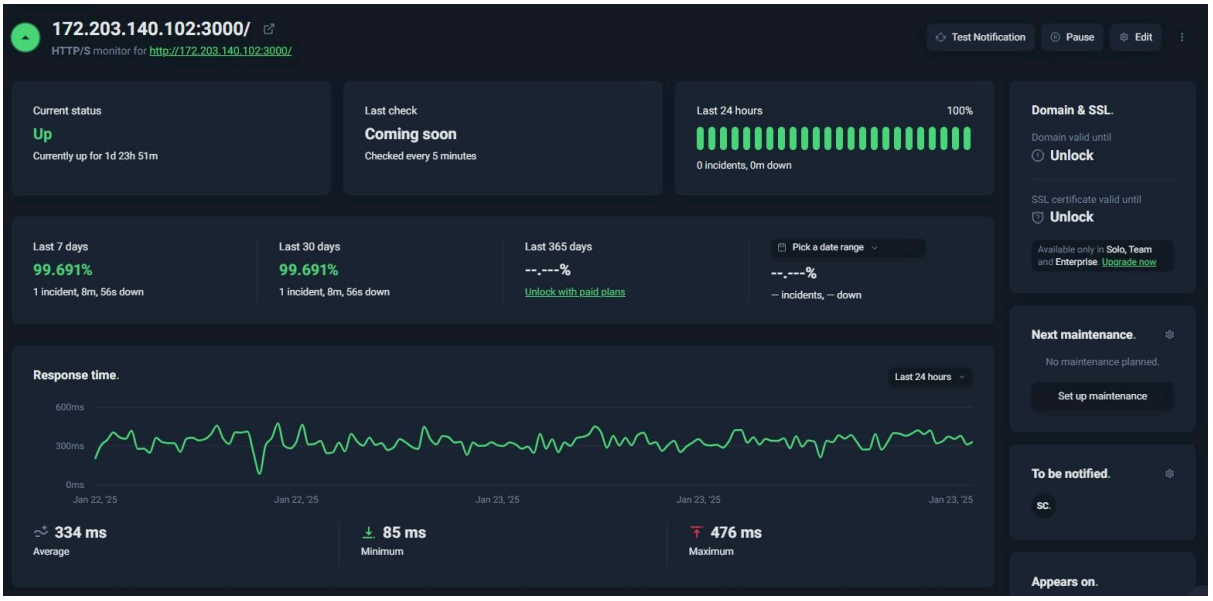

Figura 22

Resultados de pruebas con UptimeRobot después de 48 horas, con un Uptime del 99.691%

2. ***Usuarios concurrentes:***

Herramienta JMeter: https://jmeter.apache.org/

Configuración de la herramienta

Para la ejecución de las pruebas, se utilizó una máquina con Linux. Se configuró primero la prueba mediante la interfaz gráfica de JMeter, la cual se accede dirigiéndose al directorio de instalación y buscando el archivo binario de JMeter. Luego, se abre ese directorio en consola y se ejecuta el siguiente comando:

./jmeter

Hecho esto, tendremos acceso a la interfaz gráfica

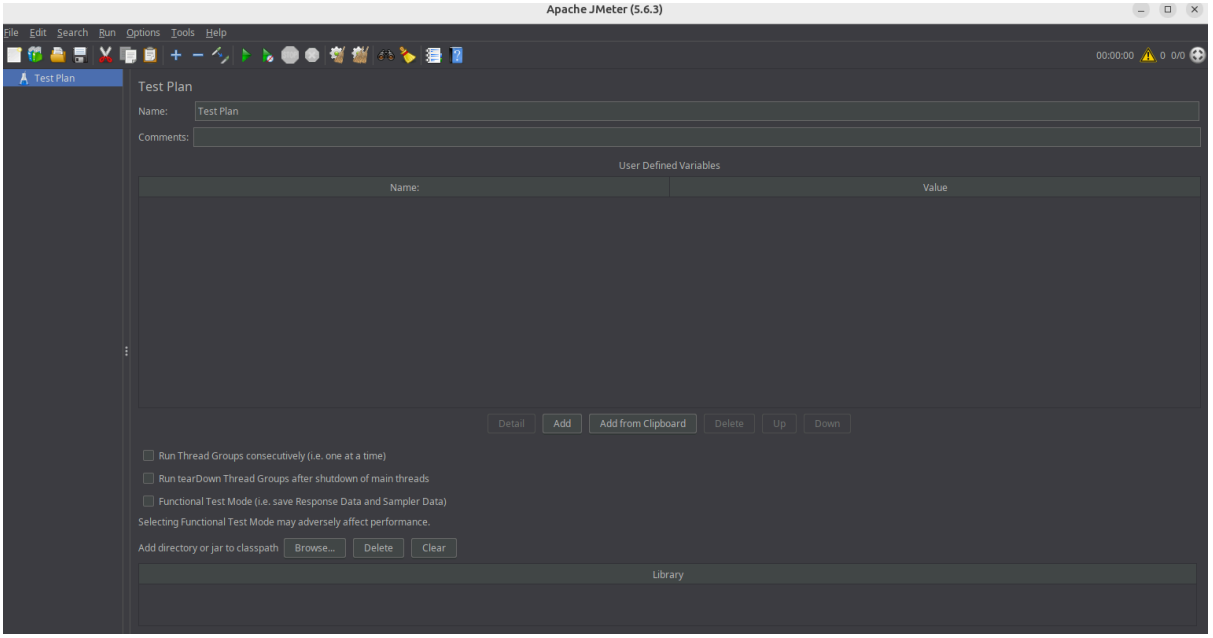

Figura 23
Interfaz gráfica de JMeter

Seleccionamos el “Test Plan” y agregamos un “Thread group”

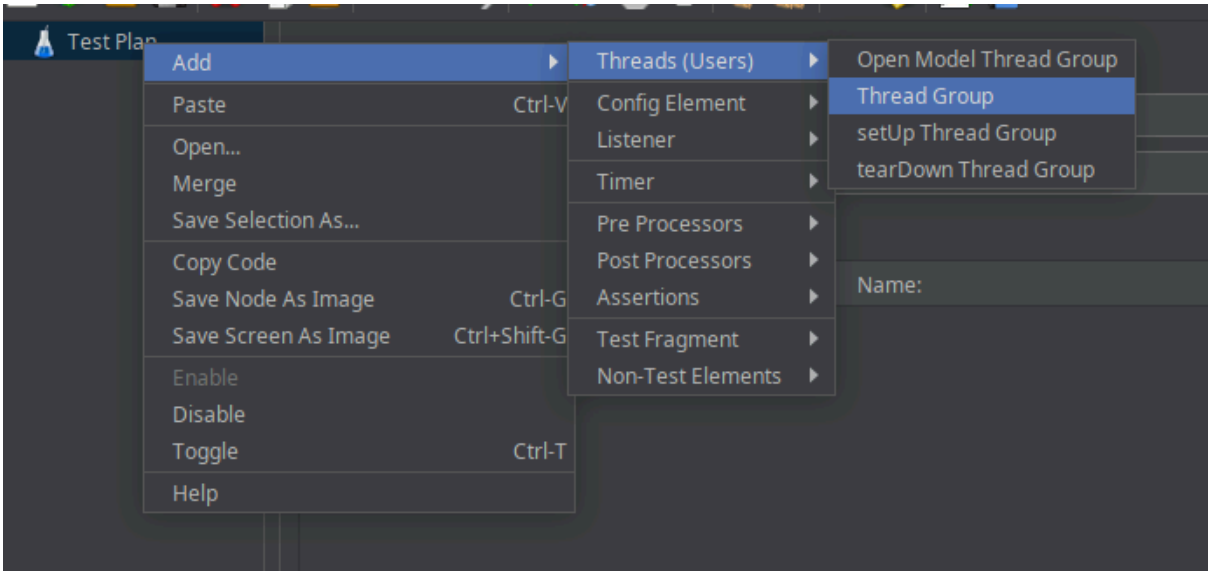

Figura 24
Agregar un Thread Group

Luego, configuramos el Thread group con un número de usuarios, un ramp up de 2 segundos y solo 1 loop.

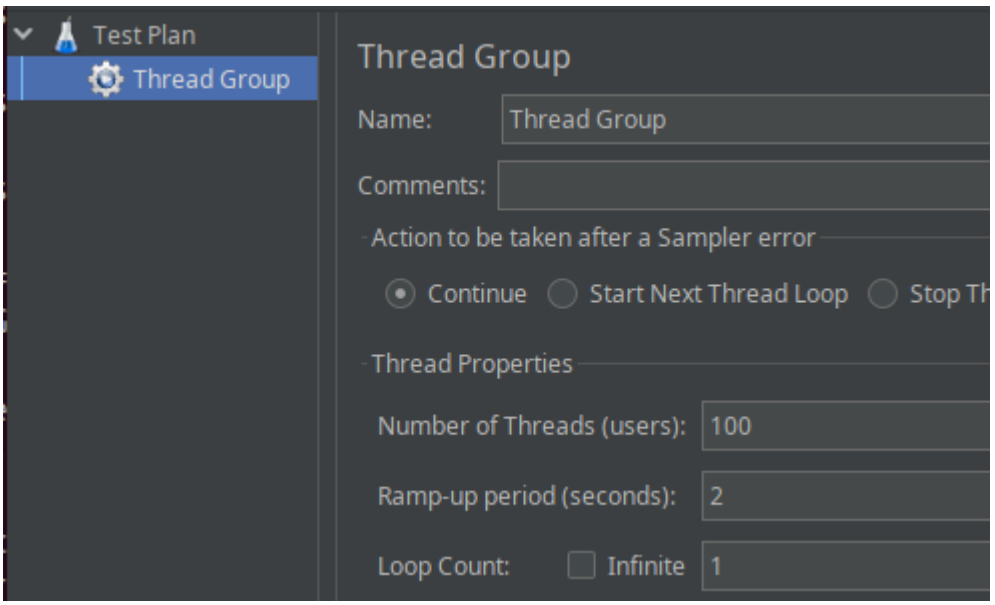

Figura 25
Configuración del Thread Group

Luego, agregamos una HTTP Request al Thread Group

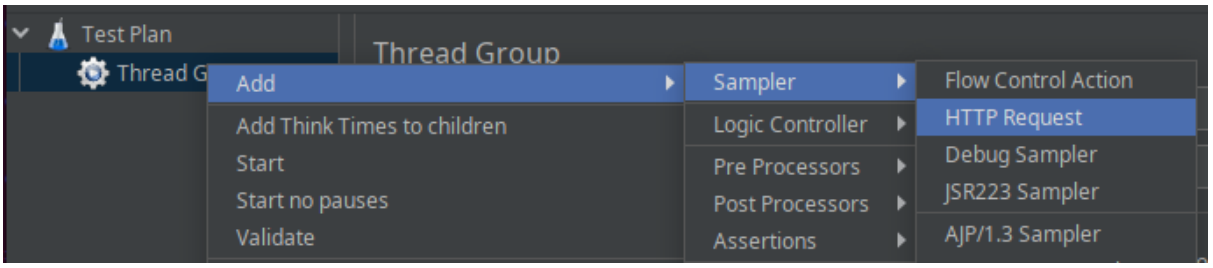

Figura 26
Agregar una HTTP Request

Luego, configuramos la solicitud HTTP para que la haga a nuestra página

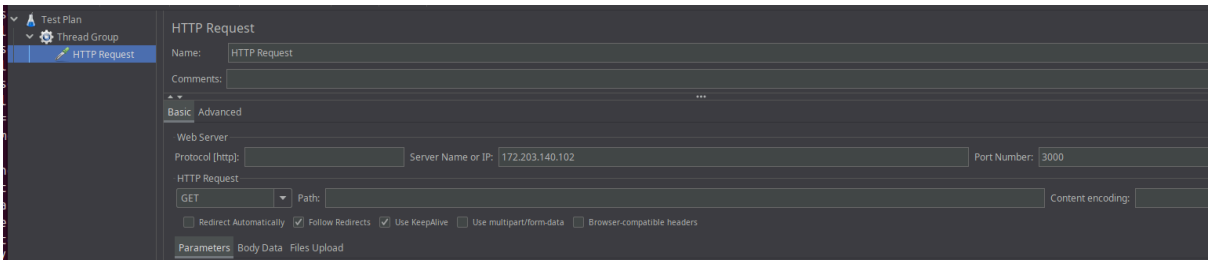

Figura 27
Configuración de la solicitud HTTP

Con esto, se guarda el archivo .jmx, se pasa a la interfaz de consola y se ejecuta el siguiente comando
/jmeter -n -t /home/sergio/Downloads/apache-jmeter-5.6.3/bin/Prueba100Usuariosjmx.jmx -l /home/sergio/Desktop/Resultado/resultado.jtl -e -o /home/sergio/Desktop/Reportes

Los dos directorios del final no importan, mientras estén vacíos. Para cada prueba, simplemente se cambia el archivo .jmx que se utiliza para la prueba, los cuales tienen diferentes números de usuarios

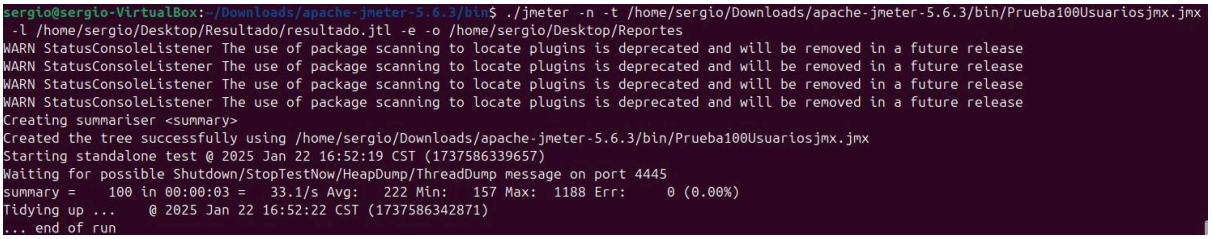

Figura 28
Resultados de prueba de 100 usuarios con JMeter.

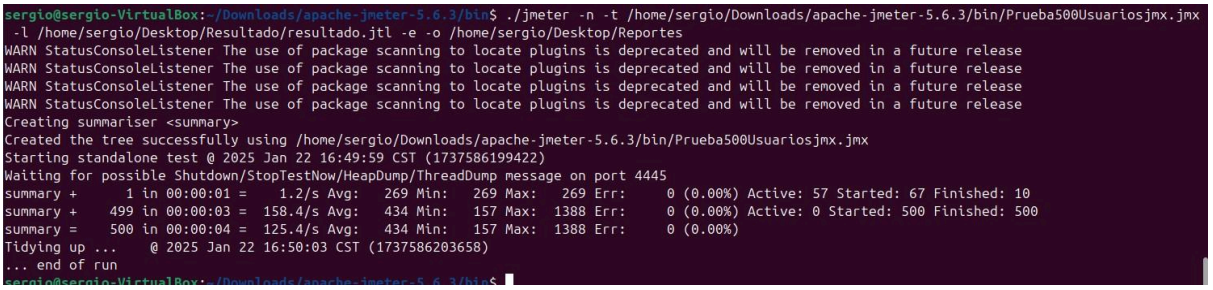

Figura 29
Resultados de prueba de 500 usuarios con JMeter.

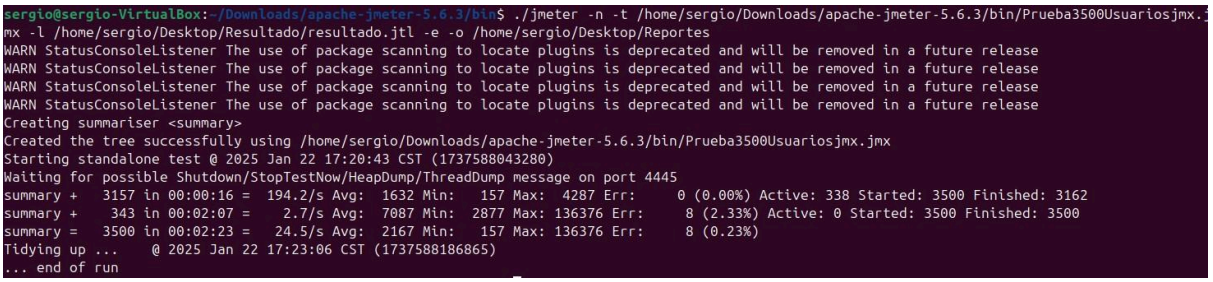

Figura 30
Resultados de prueba de 3500 usuarios con JMeter.

Pruebas de aceptación de usuario

1. ***Porcentaje de conformidad con WCAG:***

Prueba utilizando Lighthouse para medir la accesibilidad del sitio web, en esta prueba se evalúan 21 aspectos de accesibilidad.

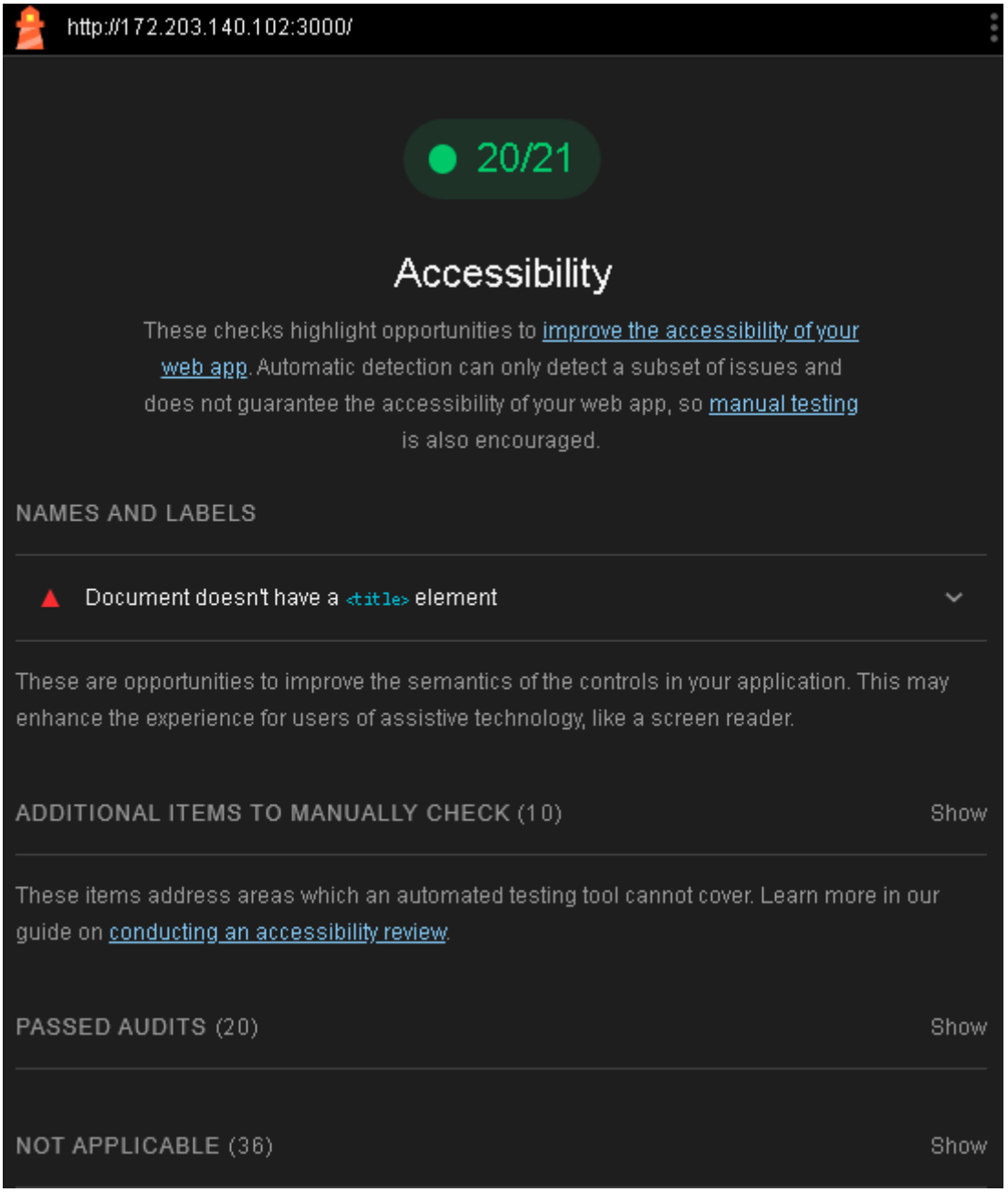

Figura 31
Resultados de pruebas con Lighthouse.

2. ***Porcentaje de Aceptación de Usuario:***

Se identificaron 3 tareas principales que podían generar disconformidad o desagrado al usuario a la hora de utilizar el sistema.

Se generó este instrumento con instrucciones a seguir para realizar las tareas y se envió a 11 usuarios de prueba para que las realizaran: https://drive.google.com/file/d/1HHXIZA4LO_iDsTOhQSfTty1aeVYsyNeq/view?usp=drive_link

Al final del documento se le pide a los usuarios ingresar a un forms (https://forms.gle/yGbGxbycRaCTDqUP8) para conocer su opinión en base a las tareas que realizaron, estas preguntas fueron desarrolladas en base a la siguiente tabla:

Tarea	Módulo	Requerimiento	Atributo
Tarea 1	Clientes	RF002 - Detalle de Cliente.	Usability - Learnability Functional Suitability - Functional Completeness
Tarea 2	Proveedores	RF003 - Consulta y Filtros de Proveedores.	Usability - User Error Protection.
Tarea 3	Ventas	RF007 - Consulta y Filtros de Ventas.	Functional Suitability - Functional Correctness.

Se tomó la moda de respuesta en cada pregunta, en base a eso se le asignó un estado de "Aprobado" o "Rechazado" a cada pregunta. De ahí se sacó el porcentaje de cada criterio sobre el total y en base al criterio de aceptación de la prueba se determinó lo siguiente:

	Criterio	Porcentaje General	Criterio de la Prueba
Resultados	APROBADO	76,92%	Más del 50% de aceptación
	RECHAZADO	23,08%	**APROBADO**

Para un proceso más detallado, consulte el manual desarrollado por Carvajal Oreamuno, V. M., et al., disponible en el apéndice de este documento (Hoja P10).

5. Resultados Obtenidos

Desglose y detalle de los resultados obtenidos en la ejecución del plan de pruebas aplicado:

Para detalles sobre el desglose de los resultados, consulte Carvajal Oreamuno, V. M. et al. (2025), disponible en: https://docs.google.com/spreadsheets/d/1IuMxKQJGfdhtimcRne_FxbeYExO7x2VcmLrjNflUNmw/edit?gid=0#gid

6. Detalle del análisis aplicado a los resultados obtenidos

Pruebas Estáticas

En la complejidad ciclomática, mostró un resultado de 50, lo cual es mucho mayor a lo esperado. Para ser aceptado, el resultado debía ser menor a 20, como el resultado fue casi el doble de lo esperado, se catalogó como "Rechazado". RF001 - Maintainability-Modifiability: Respecto a la tasa de duplicación de código, se esperaba un porcentaje menor a 5%. La prueba mostró un porcentaje de 0% en la tasa de duplicación de código, por lo tanto, el resultado fue catalogado como "Aprobado".

Respecto a los atributos. La reusabilidad mide la capacidad de un sistema, componente o módulo de software para ser reutilizado en diferentes contextos o aplicaciones, sin necesidad de realizar modificaciones significativas. La complejidad ciclomática mide la complejidad lógica de un código basado en el número de caminos independientes a partir de su código fuente. Una alta complejidad ciclomática, indica que el código será difícil de entender por su longitud, será más propenso a errores debido a la cantidad de puntos de decisión, tendrá poca modularidad y será más difícil de extender y mantener. Todo esto, refleja un bajo grado de reusabilidad. Por otro lado, la modificabilidad mide la facilidad con la que un sistema de software puede adaptarse a cambios, ya sea para corregir defectos, mejorar funcionalidades, o ajustar el sistema a nuevos requisitos. La tasa de duplicación de código se refiere al porcentaje de código fuente que se repite en diferentes partes de un sistema. Este indicador es importante en el contexto de la modificabilidad, ya que una alta duplicación de código puede dificultar y encarecer el mantenimiento y la evolución del software. En este caso, se ve que el código no es reusable, debido a que tiene una alta complejidad ciclomática. Sin embargo, sí es un código modificable, ya que no repite partes de código en diferentes partes del sistema.

Los requerimientos evaluados fueron, para la reusabilidad, todo el sistema; y para la modificabilidad, el "RF001-Consulta y Filtros de Clientes". De estas pruebas se puede concluir que a lo largo de todo el código del backend hay una alta complejidad ciclomática, lo que indica que no será un código fácil de reusar en caso de ser necesario a futuro. También, se puede ver que para el módulo de consulta y filtros de clientes, será fácil agregar nuevas funciones o modificar las existentes, ya que tiene una tasa nula de duplicación de código, lo que indica que tiene modularidad y modificabilidad.

Pruebas Unitarias

A nivel de prueba, se evaluaron dos aspectos clave: la correctitud funcional (Functional Correctness) y la protección contra errores del usuario (User Error Protection). Para el primer caso, la prueba reflejó que la aplicación no se comporta de la manera esperada, ya que los datos enviados desde el backend se presentan en varias líneas en lugar de una sola, lo que genera inconsistencias en los resultados. En escenarios donde se esperaba un único resultado, el backend devolvía hasta tres resultados diferentes. Esto llevó a catalogar la

prueba como “Fallida”, ya que no se logró el comportamiento esperado. En cuanto a la protección contra errores del usuario, el análisis mostró que el sistema no maneja adecuadamente los casos en los que se ingresan datos incorrectos en el módulo de Ventas, al no generar mensajes de error claros ni excepciones controladas, lo que también resultó en una prueba catalogada como “Rechazada”.

A nivel de atributo, la correctitud funcional (Functional Correctness) mide qué tan bien las funciones implementadas se comportan según las especificaciones. En este caso, las inconsistencias en los datos enviados desde el backend impiden que la tarea cumpla con su funcionalidad de manera adecuada, resultando en una evaluación fallida para este atributo. Por otro lado, la protección contra errores del usuario (User Error Protection) evalúa la capacidad del sistema para anticipar y mitigar errores comunes. Al no manejar correctamente las excepciones ni proporcionar mensajes de error informativos, el sistema afecta negativamente la experiencia del usuario y refleja un bajo desempeño en este atributo.

A nivel de requerimiento, el RF009 - Análisis de Compras por Proveedor y Categoría cumple parcialmente dependiendo de la cantidad de datos solicitados. Si se busca información sobre un solo proveedor, los resultados son correctos, mostrando compras mínimas, máximas y promedios. Sin embargo, cuando se solicitan datos de múltiples proveedores, el promedio calculado es general y no por proveedor, lo que invalida su utilidad para el requerimiento. Por su parte, el RF007 - Consulta y Filtros de Ventas no cumple con las métricas establecidas, ya que el módulo requiere mejoras significativas para garantizar la protección adecuada contra errores y excepciones, lo que afecta directamente la usabilidad del sistema.

Pruebas de Integración

A nivel de prueba, se realizaron dos evaluaciones principales para verificar la funcionalidad y la interoperabilidad del sistema. En el caso del RF009 - Functional Correctness, se llevaron a cabo tres pruebas específicas que analizaron las interacciones entre la API, el controlador y la base de datos. Los resultados mostraron que no todas las interacciones entre estos componentes funcionaron según lo esperado, ya que bajo ciertas condiciones de prueba no se devolvieron las respuestas correctas, lo que llevó a catalogar esta prueba como “Fallida”. Por otro lado, para el RF007 - Compatibility-Interoperability, la evaluación se centró en verificar la intercambiabilidad de formatos de datos entre los componentes del sistema. Los resultados fueron satisfactorios, ya que no se detectaron fallos, y se confirmó que los formatos de datos se procesan e intercambian correctamente entre los módulos.

Los detalles y resultados de estas pruebas pueden observarse en las figuras del punto 4, 13 al 17.

A nivel de atributo, la correctitud funcional (Functional Correctness) mide el grado en que las funciones de un sistema operan correctamente según lo especificado. En este caso, el análisis reveló que un número significativo de funciones evaluadas no mostraron el desempeño esperado, indicando que el sistema no cumple consistentemente con este atributo. En contraste, la interoperabilidad (Compatibility-Interoperability) mide la capacidad del

sistema para interactuar correctamente con otros módulos o sistemas mediante el intercambio de formatos de datos. El sistema cumplió adecuadamente con este atributo, demostrando que es capaz de manejar múltiples formatos de datos de manera eficiente y sin errores.

A nivel de requerimiento, el RF009 - Análisis de Compras por Proveedor y Categoría no cumple con una métrica fundamental para la solución del requerimiento, lo que genera insatisfacción en el análisis y procesamiento de compras en el sistema. Esto sugiere que se necesitan mejoras significativas para garantizar el correcto funcionamiento de esta funcionalidad. Por otro lado, el RF007 - Consulta y Filtros de Ventas cumple con las métricas establecidas, lo que demuestra que la funcionalidad de consulta y filtros de ventas se está implementando adecuadamente, permitiendo una evolución positiva del sistema en este aspecto.

Pruebas de Sistema

En las pruebas de sistema se buscó medir el atributo de disponibilidad y el atributo de capacidad.

A nivel de prueba, para el atributo de disponibilidad, se utilizó la herramienta de UptimeRobot, la cual proporcionaba monitoreo en tiempo real del sistema, registrando cada caída del sistema. Tras 48 horas, la herramienta proporcionó un 99.691% de tiempo de actividad, lo cual sobrepasó las expectativas, ya que, por la naturaleza del servicio de hosteo, no se esperaba que la página tuviese tan alta disponibilidad. Al superar el 99% de tiempo de actividad, la prueba se catalogó como aprobada. Para el atributo de capacidad, se utilizó la herramienta JMeter, la cual puede simular varios hilos (usuarios) realizando una conexión a la página. Tras hacer pruebas con 100, 500 y 3500 usuarios, todas las pruebas superaron el 99% de tasa de aceptación de solicitudes, con las primeras dos pruebas aceptando el 100% de las solicitudes, lo que hace que la prueba termine siendo aceptada.

A nivel de atributo, la disponibilidad mide la capacidad del sistema de estar operativo cuando se requiere. Para medir la disponibilidad dentro de nuestro sistema, se utilizó la métrica de tiempo de actividad (Uptime). Un alto tiempo de actividad indica que el sistema va a estar disponible cuando se requiera la gran mayoría de las veces. En cuanto a la capacidad, se utilizó la métrica de usuarios concurrentes. Esta métrica toma la cantidad de solicitudes que el sistema acepta y las solicitudes totales entrantes y con esto calcula el porcentaje de solicitudes que fueron aceptadas. Un alto porcentaje de aceptación indica que el sistema es capaz de manejar una alta cantidad de usuarios, por lo que tiene una alta capacidad. Al ver los resultados, se puede ver que el sistema tiene una alta disponibilidad, por lo que es muy probable que, si alguien ocupa usar el sistema, este lo va a poder hacer sin problemas. También se puede ver que el sistema tiene una buena capacidad, siendo capaz de aceptar la mayoría de las solicitudes con 3500 usuarios. Sin embargo, si se hicieran pruebas con más usuarios, la cantidad de solicitudes aceptadas podría empezar a reducirse considerablemente, pero para un sistema que no tenga muchos usuarios, se puede decir que estos dos atributos son muy aceptables.

A nivel de requerimiento, ambas pruebas de sistema se hicieron sobre el sistema en general. Esto debido a la naturaleza de la prueba y del sistema, ya que, o todo el sistema está

disponible o está caído. No puede una parte estar disponible y otra no. También con la parte la capacidad, todo el sistema tiene una misma capacidad, no se divide por módulos. Con esto se puede ver que todo el sistema tiene buena disponibilidad y una capacidad aceptable para un sistema pequeño como este.

Pruebas de aceptación de usuario

A nivel de prueba, se evaluaron dos grupos de atributos clave. Primero, la accesibilidad (Accessibility) fue probada mediante 21 pruebas específicas, de las cuales 20 resultaron exitosas, logrando un 95.23% de éxito. Este resultado fue considerado satisfactorio, demostrando que el sistema cumple en gran medida con las normas de accesibilidad evaluadas. Por otro lado, los atributos de facilidad de aprendizaje (Learnability), completitud funcional (Functional Completeness), protección contra errores del usuario (User Error Protection) y corrección funcional (Functional Correctness) fueron evaluados a través de tres tareas principales proporcionadas a los usuarios. Los resultados indicaron un criterio de rechazo, ya que los usuarios enfrentaron dificultades significativas debido a la falta de claridad en el sistema, funcionalidades incompletas y una insuficiente protección contra errores comunes.

A nivel de atributo, la accesibilidad (Accessibility) se destacó positivamente, ya que, aunque no fue un requisito explícito del proyecto, los principios de diseño aplicados permitieron una alta accesibilidad visual para usuarios con discapacidades. Esto refleja un cumplimiento efectivo de este atributo, incluso sin estar especificado en las métricas iniciales del proyecto. Por otro lado, la facilidad de aprendizaje (Learnability) se vio afectada negativamente debido a la ausencia de guías claras y retroalimentación útil, lo que dificultó la comprensión inicial del sistema. La completitud funcional (Functional Completeness) presentó limitaciones, ya que algunas funcionalidades esenciales no estaban implementadas. Además, la protección contra errores del usuario (User Error Protection) y la corrección funcional (Functional Correctness) mostraron debilidades importantes, comprometiendo la experiencia de los usuarios y la confiabilidad general del sistema.

A nivel de requerimiento, la accesibilidad fue evaluada en todo el sistema y se concluyó que el sitio web es ampliamente accesible, lo cual es un punto positivo para la aplicación. Sin embargo, los otros cuatro atributos (Learnability, Functional Completeness, User Error Protection y Functional Correctness) reflejaron un incumplimiento generalizado de los requerimientos definidos. Esto sugiere que el sistema, en su estado actual, no es adecuado para garantizar la aceptación de los usuarios, y se requieren mejoras significativas en las áreas mencionadas para asegurar la funcionalidad y usabilidad esperadas.

7. Conclusiones y recomendaciones sugeridas.

a. Conclusiones

- El sistema "Sistema de Gestión de Inventarios y Ventas" cumple parcialmente con los requerimientos funcionales y no funcionales evaluados. Los módulos principales, como Clientes, Proveedores, Ventas e Inventarios, demostraron un buen nivel de funcionalidad y usabilidad.
- Se identificaron defectos críticos en áreas relacionadas con la protección contra errores del usuario, especialmente en escenarios donde se introdujeron datos no válidos en los filtros de búsqueda.
- Los atributos de mantenibilidad, como la complejidad ciclomática, presentan oportunidades de mejora para facilitar futuros mantenimientos del sistema.
- El rendimiento del sistema en entornos simulados mostró comportamientos aceptables, pero no se evaluaron escenarios de alta concurrencia.

b. Recomendaciones

- Realizar pruebas de usuario más exhaustivas para identificar posibles puntos de fricción en la experiencia y optimizar el diseño de interfaz.
- Mejorar los mensajes de error y las validaciones en los formularios de búsqueda para reducir la tasa de errores del sistema.
- Refactorizar las secciones del código donde se detectó un alto nivel de complejidad ciclomática, utilizando funciones reutilizables o componentes más modulares.
- Realizar pruebas de rendimiento adicionales en entornos de producción simulados para identificar posibles problemas de escalabilidad y capacidad.

8. Bibliografía y referencias consultadas

Apache JMeter™. (s/f). Apache.org. Recuperado el 26 de enero de 2025, de https://jmeter.apache.org/

Clayton, L. (2023, octubre 25). *How to calculate uptime? And 5 tips for achieving 99.999%.* UptimeRobot Blog; UptimeRobot. https://uptimerobot.com/blog/how-to-calculate-uptime/

Code metrics. (s/f). Sonarsource.com. Recuperado el 26 de enero de 2025, de https://docs.sonarsource.com/sonarqube-server/latest/user-guide/code-metrics/metrics-definition/

Cyclomatic complexity: Understanding an essential metric. (2024, febrero 22). Metridev. https://www.metridev.com/metrics/cyclomatic-complexity-understanding-an-essential-metric/

Karanam, R. (2019, noviembre 14). Code quality basics - What Is Code Duplication? *Spring Boot Tutorial*. https://www.springboottutorial.com/code-quality-what-is-code-duplication

Montoto, O. C. (s/f). *AuditTool WCAG 2.2. Herramienta para auditorías de accesibilidad de acuerdo a las WCAG 2.2*. Blogspot.com. Recuperado el 26 de enero de 2025, de https://olgacarreras.blogspot.com/2023/12/audit-tool-wcag-22-herramienta-para.html

Systems and software engineering -- Systems and software product Quality Requirements and Evaluation (SQuaRE) -- Common Industry Format (CIF) for usability: General framework for usability-related information. (s/f). Tecnoaccesible.net. Recuperado el 26 de enero de 2025, de

https://tecnoaccesible.net/documentos/systems-and-software-engineering-systems-and-software-product-quality-requirements-and

Understanding cyclomatic complexity and its importance in software development. (2023, abril 13). BlueOptima. https://www.blueoptima.com/understanding-cyclomatic-complexity-and-its-importance-in-software-development/

UX Design books and articles. (s/f). The Interaction Design Foundation. Recuperado el 26 de enero de 2025, de https://www.interaction-design.org/literature

Yaremchuk, S., Bardis, N., & Vyacheslav, K. (2017). Metric-based method of software requirements correctness improvement. *ITM web of conferences*, *9*, 03009. https://doi.org/10.1051/itmconf/20170903009

(S/f). Nngroup.com. Recuperado el 26 de enero de 2025, de https://www.nngroup.com/articles/measuring-ux-metrics/

9. Apéndices

Carvajal Oreamuno, V. M., Calderon Perez, L. F., Chavarría Avilés, S., Calvo Rodriguez, K. Y., & Lopez Herrera, A. D. (2025). Proyectos_ComplementoPlantillaIEEE-829-19982_Verano_24-25 [Hoja de cálculo]. Recuperado de https://estudianteccr-my.sharepoint.com/:x:/g/personal/andrewlopezherrera_estudiantec_cr/Eb3ku53p7xhIvvV_utYV0nMB0muE0VsnKhn4HvWZADRG-w?rtime=eXFE2Ng73Ug

Carvajal Oreamuno, V. M., Calderon Perez, L. F., Chavarría Avilés, S., Calvo Rodriguez, K. Y., & Lopez Herrera, A. D. (2025). Desglose del resultado de las pruebas [Hoja de cálculo]. Recuperado de https://docs.google.com/spreadsheets/d/1IuMxKQJGfdhtimcRne_FxbeYExO7x2VcmLrjNflUNmw/edit?gid=0#gid=0