Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No.3, Mes Marzo, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 1-18
http://publicaciones.uci.cu

Tipo de artículo: Artículo original
Temática: soluciones informáticas
Recibido: 11/01/2019 | Aceptado: 22/02/2019 | Publicado: 20/03/2019

Implementación de pruebas de software para el módulo para el diseño de mapas de impresión para la plataforma ULTRON

Implementation of software tests for print map design module for the ULTRON platform

Yelena Vento Diaz[1], Gerdys Ernesto Jiménez Moya[2], Reinier Suarez Estevez[3]

[1] Facultad de Ciencias y Tecnologías Computacionales, Universidad de las Ciencias Informáticas, yvento@estudiantes.uci.cu
[2] Facultad de Ciencias y Tecnologías Computacionales, Universidad de las Ciencias Informáticas, gejimenez@uci.cu
[3] Facultad de Ciencias y Tecnologías Computacionales, Universidad de las Ciencias Informáticas, restevez@uci.cu

[*] Autor para correspondencia: yvento@estudiantes.uci.cu

Resumen

A partir del proceso de implementación del módulo para el diseño de mapas de impresión para la plataforma ULTRON, se hace necesario evaluar el funcionamiento de manera correcta de los requisitos especificados. Las pruebas del software son un elemento crítico para la garantía de calidad del software y representa una revisión final de las especificaciones, del diseño y de la codificación. La presente investigación describe una solución al proceso de prueba después de la implementación de los requisitos definidos para lo cual se realiza una estrategia de pruebas definida por tres niveles: unitaria, integración y sistema. Además, se describe y fundamenta el proceso de validación de la solución implementada, mediante la utilización de los casos de pruebas y el uso de las herramientas de rendimiento y funcionalidad.

Palabras claves: pruebas de software, mapas de impresión, sistemas de información geográfica.

Abstract

From the process of implementing the module for the design of printing maps for the ULTRON platform, it is necessary to evaluate the correct functioning of the specified requirements. Software testing is a critical element for software quality assurance and represents a final revision of the specifications, design and coding. The present investigation describes a solution

to the testing process after the implementation of the defined requirements for which a test strategy is carried out defined by three levels: unitary, integration and system. In addition, the validation process of the implemented solution is described and based on the use of test cases and the use of performance and functionality tools.

Keywords*:* software tests, printing maps, geographic information systems.

Introducción

La presente investigación describe los diferentes mecanismos utilizados para llevar a cabo el desarrollo y validación del módulo para el diseño de mapas de impresión para la plataforma ULTRON. Además, se procede a desarrollar el flujo de trabajo de implementación. Se describen sus principales artefactos, como el modelo de implementación, subsistemas de implementación, diagramas de componentes y el modelo de despliegue del sistema. Posteriormente se le aplican las pruebas al módulo partiendo de la confección y descripción de los casos de prueba.

Materiales y métodos o Metodología computacional

Modelo de implementación: es comprendido por un conjunto de componentes y subsistemas que constituyen la composición física de la implementación del sistema. Entre los componentes podemos encontrar datos, archivos, ejecutables, código fuente y los directorios. Fundamentalmente, se describe la relación que existe desde los paquetes y clases del modelo de diseño a subsistemas y componentes físicos (Cruz Medina, 2017). A continuación, se muestra como han quedado modelado el diagrama de componentes y modelo de despliegue del módulo.

Diagrama de componentes.

Los diagramas de componentes muestran los elementos de diseño de un sistema de información. Permiten visualizar con más facilidad la estructura general del sistema y el comportamiento del servicio que estos componentes proporcionan y utilizan a través de las interfaces. Un diagrama de componentes tiene un nivel más alto de abstracción que un diagrama de clase, usualmente un componente se implementa por una o más clases en tiempo de ejecución. (Leon, 2016). En la

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No.3, Mes Marzo, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 1-18
http://publicaciones.uci.cu

siguiente figura se muestra el diagrama de componentes asociado al módulo de diseño de mapas de impresión.

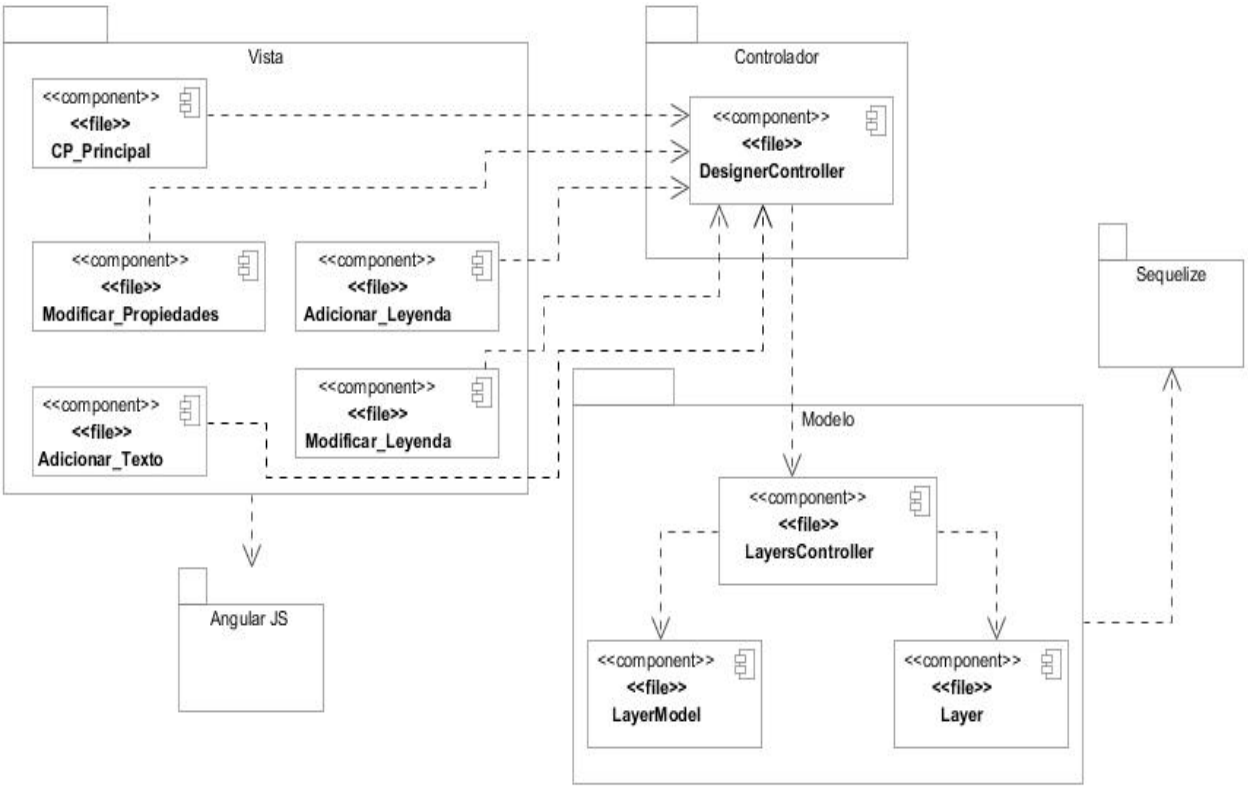

Fig. 1: Diagrama de Componente

Modelo de despliegue

Los diagramas de despliegue son diagramas del UML que representan la arquitectura física de un sistema informático. Muestran las instancias de los componentes en tiempo de ejecución y sus asociaciones. También muestran interfaces y objetos (instancias de clases). (DELGADO, 2015). En la siguiente figura se muestra el diagrama de despliegue del módulo desarrollado.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No.3, Mes Marzo, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 1-18
http://publicaciones.uci.cu

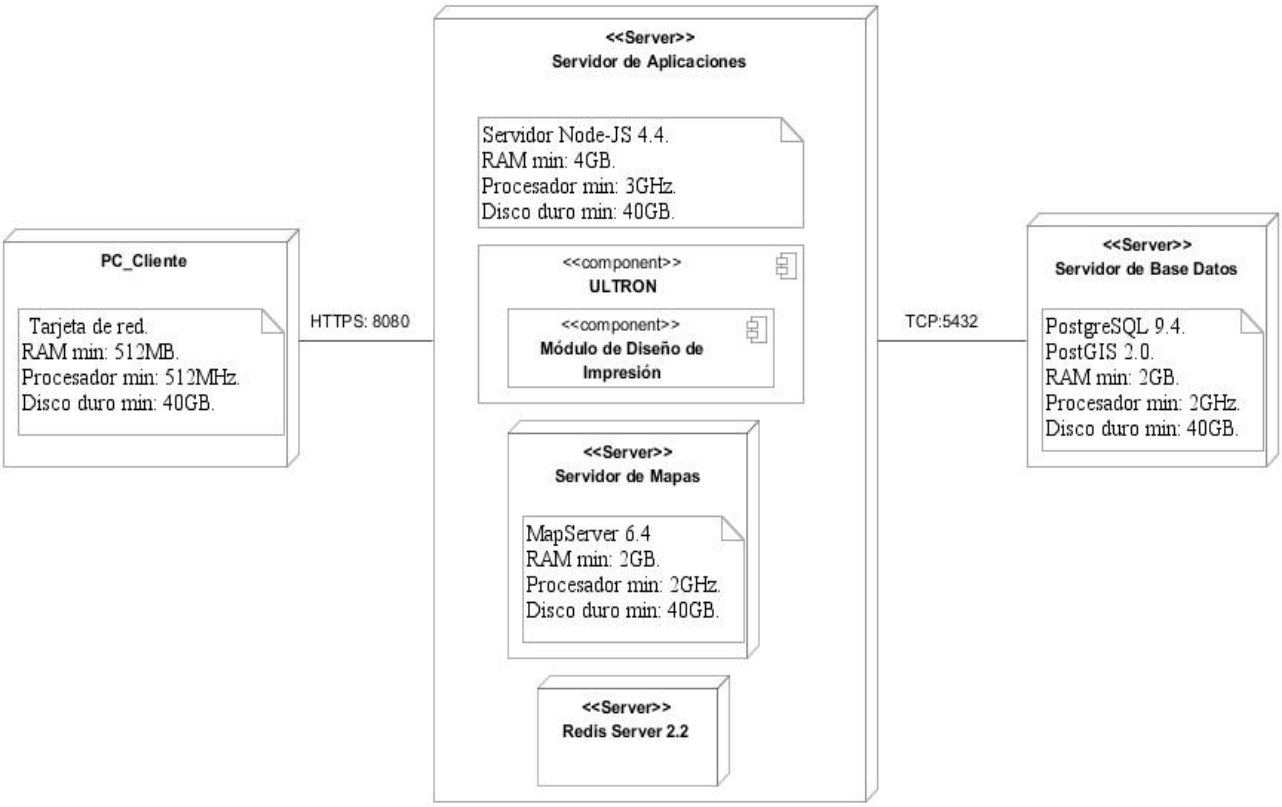

Fig. 2: Diagrama de Despliegue.

Estándares de codificación.

Los estándares de código, son parte de las llamadas buenas practicas o mejores prácticas. Estas son un conjunto no formal de reglas, que han ido surgiendo en las distintas comunidades de desarrolladores con el paso del tiempo y las cuales, bien aplicadas pueden incrementar la calidad de tu código. Algunos beneficios de la codificación estandarizada es el mejoramiento de la comunicación en los equipos de desarrollo de software, reduce los errores de programación y mejora la calidad del software. (Moreno Barrios, et al., 2015). Para la implementación del módulo se hacen uso de los siguientes estándares de codificación definidos por Jairo Rojas Delgado para el desarrollo de la plataforma ULTRON:

- Añadir comentarios para explicar determinadas partes de tu código.
- Al elegir nombres de las variables, se recomienda el utilizar variables descriptivas.
- Las funciones tienen que seguir el siguiente esquema de nombrado: modulo_submodulo_nombre_funcion ().
- Las funciones se escriben en minúsculas y separados con guiones de suelo, y los argumentos que reciben son descriptivos.
- Todo el código desarrollado tendrá una indentación de 4 espacios.

- La indentación se realizará siempre con tabuladores y no con cuatro puntos cuidando de no mezclar un estilo con el otro.
- Se dejará una línea en blanco inmediatamente después del inicio de un bloque de código, dígase bloques de funciones, bloques condicionales o bucles.
- La llave de apertura de un bloque de código se coloca después de la instrucción anterior y no en una línea en blanco independiente.
- La llave de clausura de un bloque de código tendrá el mismo nivel de indentación que la línea de la llave de apertura.
- Cada bloque de código anidado tendrá un nivel de indentación más que el bloque padre exceptuando las llaves de apertura y cierre.
- Todas las variables de un bloque de código se declararán al inicio del método en cuestión.
- Las variables se declaran una por línea y empleando la palabra reservada <var> que nunca podrá omitirse, aunque el lenguaje lo permita.

Resultados y discusiones

Para dar inicio a las pruebas de un software lo primero es describir una estrategia de prueba donde quede plasmado los niveles de prueba a tratar, así como los tipos de prueba a emplear en cada nivel, los métodos de prueba a aplicar, así como las técnicas a utilizar para cada método.

Estrategia de pruebas.

La estrategia de pruebas permite indicar los niveles y tipos de pruebas que se van a realizar en la aplicación. También detalla los módulos, componentes y funcionalidades; así como la cobertura de pruebas a aplicar en cada uno de ellos. Además, logra indicar los posibles riesgos del proyecto y las acciones para mitigarlos. (Garrido, 2017)

Niveles de pruebas.

Fueron definidos los siguientes niveles de pruebas que a continuación se describen según (Quijano, 2018)

- Pruebas Unitarias: Las pruebas unitarias son pruebas automatizadas que verifican la funcionalidad en el componente, clase, método o nivel de propiedad. El objetivo principal de estas es tomar la pieza más pequeña de software comprobable en la aplicación, aislarla del resto del código y determinar si se comporta exactamente como esperamos. Cada unidad se prueba por separado antes de integrarlas en los componentes para probar las interfaces entre las unidades.

- Pruebas de Integración: Desde una perspectiva de prueba, las unidades individuales se integran juntas para formar componentes más grandes. En su forma más simple, dos unidades que ya han sido probadas se combinan en un componente integrado y se prueba la interfaz entre ellas. Las pruebas de integración o de componentes, identifican problemas que ocurren cuando las unidades se combinan. Los nuevos errores que surgen probablemente estén relacionados con la interfaz entre las unidades en lugar de dentro de las propias unidades; simplificando la tarea de encontrar y corregir los defectos.

- Pruebas de Sistema: Las pruebas del sistema tienen como objetivo ejercitar profundamente el sistema. Estas verifican el funcionamiento correcto de las interfaces entre los distintos subsistemas o módulos que lo componen y con el resto de sistemas de información con los que se comunica. Permiten probar el sistema en su conjunto y con otros sistemas con los que se relaciona para verificar que las especificaciones funcionales y técnicas se cumplen.

Tipos de pruebas.

- Pruebas de Funcionalidad: El servicio de pruebas funcionales se centra en comprobar que los sistemas desarrollados funcionan acorde a las especificaciones funcionales y requisitos del cliente. Este servicio ayuda a su organización a detectar los posibles defectos derivados de errores en la fase de programación. Las características no funcionales del software se pueden medir de diversas maneras, por ejemplo, por medio de tiempos de respuesta en el caso de pruebas de rendimiento o por número máximo de sesiones en pruebas de estrés. (Panel Testing, 2015)

- Pruebas no Funcionales: Se enfocan en validar un sistema o aplicación por medio de sus requerimientos no funcionales, la forma en que el sistema funciona y no por medio de comportamientos específicos. Estas permiten conocer qué riesgos corre el producto y nos dicen si tiene un mal desempeño o un bajo rendimiento en los entornos de producción. Permiten explicar lo que soporta el producto y si cumple con las expectativas de los clientes. (Morales, 2018)

    - Pruebas de Carga: Consiste en someter al sistema a altas cargas de trabajo, simulando la actividad real de los futuros usuarios del sistema. Estas pruebas nos ayudarán a predecir el comportamiento de nuestro sistema y conocer el grado en que realiza las funciones para las que ha sido diseñado, sin pérdidas de servicio y con un tiempo de respuesta óptimo y estable. (Ormaetxea, 2016)

- Pruebas de Usabilidad: Se definen como la práctica de probar un producto físico o digital con usuarios representativos, para identificar áreas de mejora en el desempeño y la experiencia de uso. Estas pruebas se hacen a una aplicación o un sitio web con el fin de comprobar la comodidad, facilidad o complejidad con la que se maneja. Son centradas en el usuario observando detalladamente su reacción frente al diseño y funcionalidad del producto a evaluar. (Cantú, 2017)

Métodos de pruebas.

- Método de Caja Negra: Estas pruebas son realizadas desde la interfaz gráfica, es un método de pruebas de software en el cual la funcionalidad se verifica sin tomar en cuenta la estructura interna de código, detalles de implementación o escenarios de ejecución internos en el software. Se enfoca solamente en las entradas y salidas del sistema, apoyándose en la especificación de requisitos y documentación funcional. (Terrera, 2017). Para la realización de este método se utiliza la técnica de Partición de Equivalencia.
- Método de Usuarios Expertos: Los usuarios expertos contribuyen a las pruebas de usabilidad detectando errores del sistema, basando sus opiniones en su propia experiencia. (Ruis Eguino, et al., 2013). Para la realización de este método se optó por la utilización de la técnica de Evaluaciones Heurísticas.

Técnicas de pruebas.

- Partición de Equivalencia: Esta técnica consiste en clasificar las entradas de datos del sistema en grupos que presentan un comportamiento similar. Las particiones también pueden definirse en función de las salidas de datos, valores internos, valores relacionados antes o después de ciertos eventos, y también para los valores que reciben las interfaces. Además, se definen pruebas para cubrir todos o parte de las particiones de datos válidos y datos inválidos. Es aplicable a entradas de datos realizadas por personas o vía interfaces con otros sistemas. (Terrera, 2017)
- Evaluaciones heurísticas: Una evaluación heurística es un método de inspección de usabilidad de software que ayuda a identificar problemas en la interfaz de usuario. Supone que los evaluadores examinan y juzgan su conformidad con los principios reconocidos los cuales son llamados también heurísticas de usabilidad. Estas generalmente se llevan a cabo por un pequeño grupo (de uno a tres) de probadores, los cuales de forma independiente examinan una interfaz de usuario y juzgan el cumplimiento de un conjunto de principios de usabilidad. (Prieto, et al., 2013)

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No.3, Mes Marzo, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 1-18
http://publicaciones.uci.cu

Aplicación y resultados de las pruebas.

Para llevar a cabo las pruebas unitarias se utilizó la herramienta QUnit v1.11.0 la cual basa su funcionamiento en una clase test que posibilita probar el código de aplicaciones web desarrolladas en JavaScript. Estas fueron desarrolladas constantemente cada vez que se implementaba alguna de las funcionalidades, lo que posibilitó una correcta implementación de todos los requisitos del módulo. A continuación, se muestran los resultados de las pruebas unitarias asociadas a los métodos de los requisitos: editar propiedades del documento y modificar leyenda.

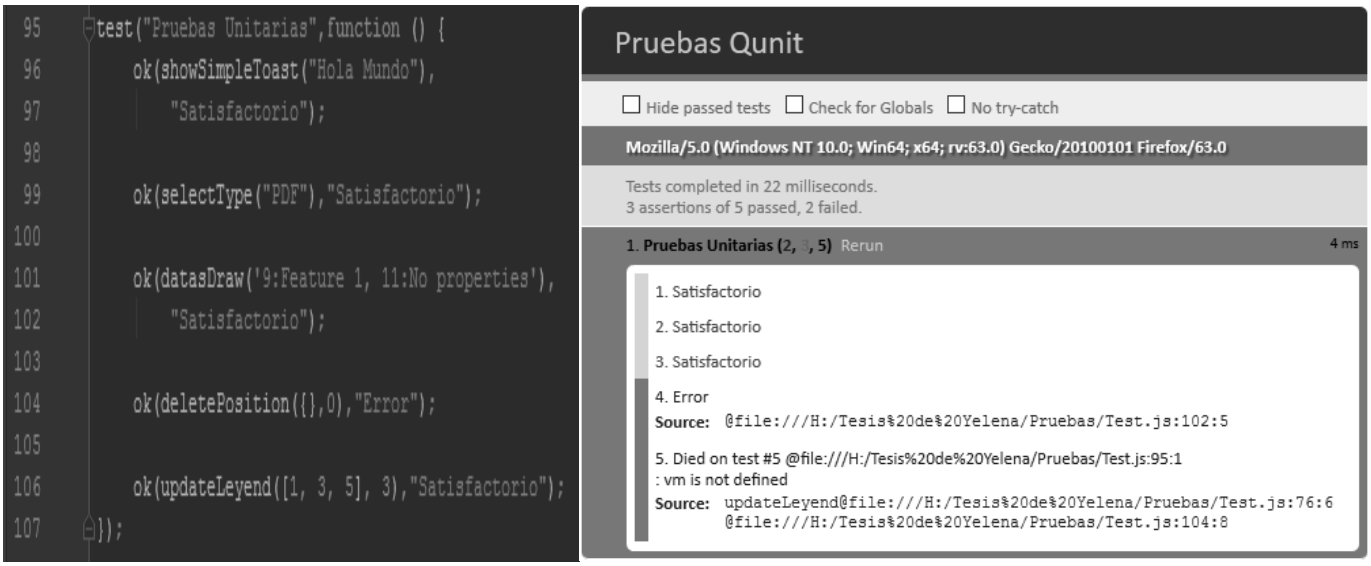

Fig. 3: Ejemplo de prueba unitaria.

Una vez finalizada dicha prueba, fueron solucionados los errores detectados, y repetido este proceso de manera cíclica hasta no encontrar no conformidades en el código de la solución propuesta. Posteriormente, se hace necesaria la realización de pruebas de integración. Estas se realizaron con la finalidad de validar la compatibilidad y el funcionamiento de las interfaces que comunican las diferentes partes que componen la solución informática. Para su realización se probó el módulo en la plataforma ULTRON, empleando el método de integración ascendente, ejecutándose correctamente todas las funcionalidades del mismo. Además, se mostraron satisfactoriamente todas las vistas asociadas a la información deseada. El desarrollo de las pruebas garantizó que el módulo diseño de mapas de impresión presentaba un correcto funcionamiento una vez integrado al sistema.

Con el fin de comprobar la usabilidad del sistema, se empleó las pruebas con usuarios expertos basada en la técnica evaluaciones heurísticas. Fueron seleccionados 3 especialistas del proyecto

(analista y dos desarrolladores). Estos especialistas interactuaron con el sistema y ejecutaron las funcionalidades para lo cual fue diseñado, con el fin de evaluar la interfaz a través de las heurísticas de diseño descritas por Jakob Nielsen, en su libro Usability Engineering (Rojas Rodríguez, et al., 2017). El equipo de desarrollo fue tomando nota de los errores que tenía el sistema en función de sus funcionalidades. Estas heurísticas estuvieron definidas por 10 criterios dados por dicho autor, las cuales se muestran a continuación:

Tabla 1: Heurísticas de usabilidad.

ID	Principio	Definición
1	Visibilidad del sistema	El sistema debe mantener siempre informado al usuario sobre lo que está ocurriendo, a través de retroalimentación dentro de un tiempo razonable.
2	Coincidencia entre el sistema y el mundo real	El sistema debería "hablar" el idioma del usuario, con palabras, frases y conceptos familiares, más que términos orientados al sistema. Seguir convenciones, de modo que la información parezca lógica y natural.
3	Control y libertad del usuario	Los usuarios eligen a veces funciones del sistema por error, y necesitarán una salida de emergencia, con opciones de deshacer y rehacer.
4	Consistencia y estándares	Los usuarios no deberían preguntarse si diferentes palabras, situaciones o acciones significan lo mismo. Se deben seguir las convenciones de la plataforma.
5	Prevención de errores	Mucho más adecuado que mostrar mensajes de error entendibles, es un diseño cuidadoso que evite la ocurrencia de problemas. Se deben eliminar estas situaciones presentando una opción de confirmación a los usuarios antes de que realicen la acción.
6	Minimizar la carga de memoria	Reconocimientos, más que recordatorios. Minimizar la carga de memoria de los usuarios mediante el uso de objetivos, acciones y opciones visibles. El usuario no debería recordar información de una parte del sistema a otra.
7	Flexibilidad y eficacia del uso	Aceleradores que pasan desapercibidos para los usuarios novatos, pero que deben agilizar la interacción con el sistema a los usuarios expertos. Debe facilitar a los usuarios la ejecución de acciones frecuentes.
8	Diseño estético y minimalista	El sistema no debe mostrar información que sea poco relevante o que rara vez sea utilizada por el usuario.
9	Ayuda al usuario para reconocer, diagnosticar y recuperarse de errores	Los mensajes de error deben estar expresados en un lenguaje natural entendible a los usuarios (no en código o lenguaje máquina). Estos deben indicar de manera precisa el problema y sugerir una solución de forma constructiva.
10	Ayuda y documentación	Cualquier tipo de información debe ser fácil de buscar y estar centrado en la tarea del usuario. Las instrucciones deben consistir en una lista concreta no muy extensa de tareas a realizar.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No.3, Mes Marzo, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 1-18
http://publicaciones.uci.cu

Para llevar a cabo dicha prueba se ha establecido una lista de chequeo con un conjunto de requerimientos formulados en las 10 categorías. A continuación, se muestran los ejemplos para 2 de ellas:

Forma de uso “Evaluación”: El mismo se evalúa de 0 en caso de mal (cuando la respuesta al indicador sea “No”) y 1 en caso que elemento revisado no presente errores (cuando la respuesta al indicador sea “Sí”).

- Visibilidad del sistema
  - ✓ Mantiene la homogeneidad de estilo con el resto de elementos del sitio web.
  - ✓ Se ubica en los lugares preestablecidos, sin romper con la composición estándar.
  - ✓ Las etiquetas de los menús son descriptivas de cada una de las opciones.
  - ✓ No incluye más de siete opciones, o si lo hace, existen subcategorías.
  - ✓ Favorecen la previsibilidad, usando términos que anticipen al usuario lo que se encontrará detrás.
  - ✓ Su longitud se adecua a la disponible en cada caso (según se trate de un menú o una cabecera).
  - ✓ Tienen un tratamiento gráfico coherente según el tipo de etiqueta: color, tamaño y tipografía adecuados.
  - ✓ Mantiene la alineación entre los campos para asegurar la armonía visual.
  - ✓ Diferencia visualmente los campos que son de carácter obligatorio.

- Diseño estético y minimalista
  - ✓ Los elementos (bloques de texto, imágenes, tablas) quedan alineados con las líneas que crean los elementos estructurales de la página: líneas de cabecera, logotipo o menús.
  - ✓ La composición de los elementos responde a formas rectangulares apaisadas, pues son las que más armonía generan y más se adaptan a las líneas maestras del monitor.
  - ✓ El orden de los elementos se ajusta a una línea visual que va de izquierda a derecha y de arriba a abajo.
  - ✓ Los elementos se sitúan en la línea visual de forma jerárquica (cuanto más abarcan, antes se encuentran).

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No.3, Mes Marzo, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 1-18
http://publicaciones.uci.cu

- ✓ Los elementos visuales (texto, tablas o gráficos) se adecuan a la paleta de colores establecida para el sitio web.
- ✓ Hay suficiente contraste entre el color del texto y el color de fondo, tanto en textos convencionales como dentro de tablas o diagramas.
- ✓ Los íconos utilizados evocan lo representado sin lugar a equívocos ni lecturas ambiguas.
- ✓ Guardan coherencia y homogeneidad con el resto de iconos del sitio web.
- ✓ Se usan fondos de color blanco o tonos tenues (gris, crema, azul pastel) y nunca de colores vivos.
- ✓ Si se usan colores para representar datos u otras informaciones en diagramas, no se contrastan verdes sobre rojos o marrones, pues es la forma de daltonismo más habitual.

Luego de realizada la prueba, apoyándose en los cuestionarios anteriores se puede apreciar que, de un total de 19 indicadores de usabilidad, el módulo implementado cumple con 18 indicadores, cifra que representa el 95% de usabilidad para las funciones presentes. Para una mejor comprensión de los resultados obtenidos se han representado los mismos en la siguiente gráfica.

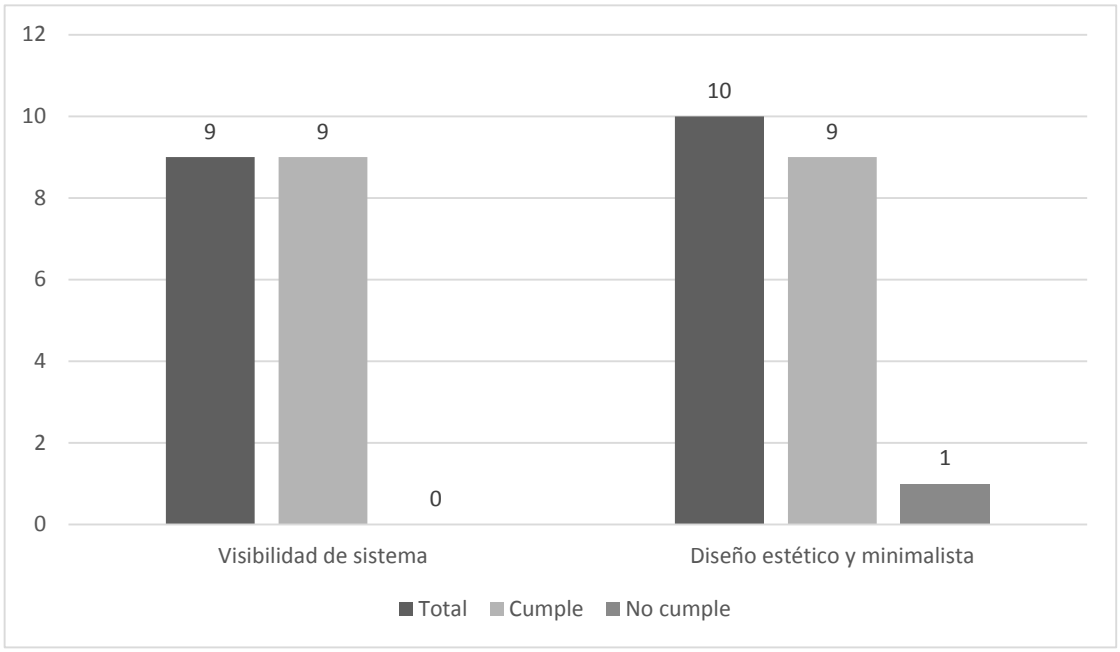

Fig. 4: Cumplimiento de los indicadores de las categorías de usabilidad.

Los resultados obtenidos fueron satisfactorios ya que los especialistas manifestaron que el módulo presentaba una interfaz sencilla e intuitiva. Por último, los expertos involucrados en la prueba expresaron el estar satisfechos con el diseño del módulo.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No.3, Mes Marzo, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 1-18
http://publicaciones.uci.cu

Para la realización de las pruebas de carga se empleó la herramienta Apache JMeter. El ambiente de prueba estuvo conformado por:

- Sistema Operativo: Windows 10
- Microprocesador: AMD E-300 @1.30 GHz
- Memoria RAM: 4 GB
- Disco Duro: 350 GB

Los resultados de las pruebas de carga se consideran satisfactorios. Esto es debido a que los tiempos de respuesta del servidor ante la interacción de 20, 60 y 100 usuarios concurrentes se encontraron en el rango de tiempo de 1 a 6 segundos. Para ello fueron utilizados las principales funciones del módulo con un volumen de carga marcado por geometrías de hasta los 1000 vértices. Con ello queda demostrado que la propuesta de solución es estable, ya que se mantuvo prestando servicios todo el tiempo, sin incurrir en fallos.

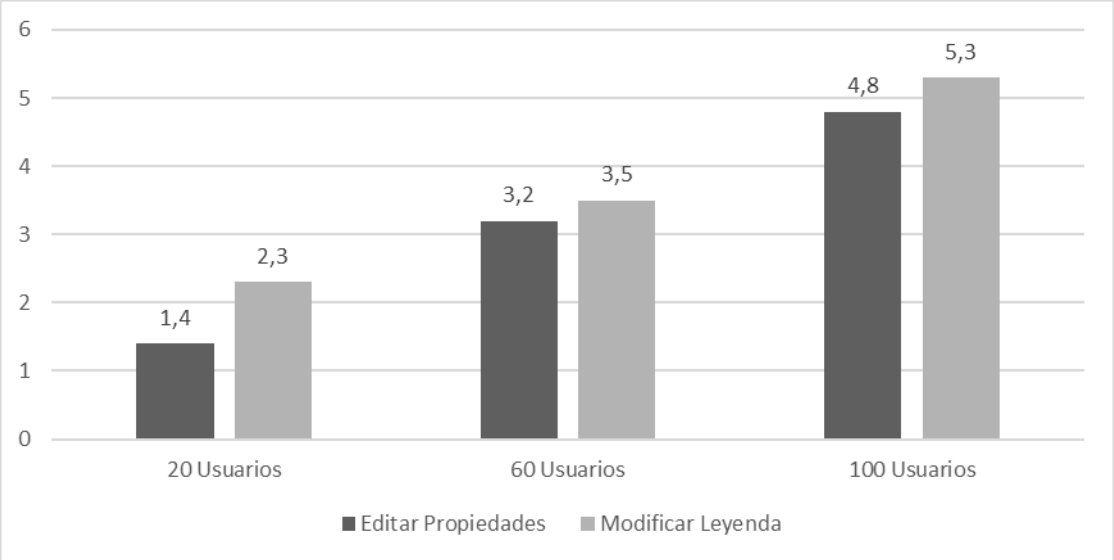

Fig. 5: Resultados de la prueba de carga para usuarios concurrentes.

Las pruebas de caja negra o funcionales son realizadas a la interfaz del software y aplicadas a las funcionalidades del mismo. Como parte del proceso de dichas pruebas al módulo se diseñaron los casos de prueba en correspondencia con las descripciones de las historias de usuario. A continuación, se presentan la Descripción del Caso de Prueba correspondiente a los requisitos Editar propiedades del documento y Adicionar etiqueta de texto.

Tabla 2: Descripción de variables del caso de prueba correspondiente a los requisitos Editar propiedades del documento y Adicionar etiqueta de texto.

No.	Nombre del Campo	Clasificación	Valor Nulo	Descripción
1	Título	Campo de texto	No	Debe introducirse cualquier combinación alfanumérica menor de 50 caracteres.
2	Tipo de documento	Lista desplegable	No	Selección única de un listado de tipos de documento.
3	Tamaño de página	Lista desplegable	No	Selección única de un listado de tamaño de página.
4	Resolución	Lista desplegable	No	Selección única de un listado de resoluciones.
5	Nota de texto	Campo de texto	Si	Debe introducirse cualquier combinación alfanumérica menor de 250 caracteres.

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No.3, Mes Marzo, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 1-18
http://publicaciones.uci.cu

Tabla 3: Descripción del Caso de Prueba para los RF: Editar propiedades del documento y Adicionar etiqueta de texto.

Escenario	Descripción	Título	Tipo de documento	Tamaño de página	Resolución	Nota de texto	Respuesta del sistema	Flujo Central
EC 1 Editar propiedades del documento correctamente	El usuario introduce todos los parámetros correctamente para editar propiedades del documento	V Mapa1	V PDF	V A2	V 72 dpi	V Nuevo mapa de impresión	Modifica las propiedades de documento de impresión.	Se selecciona la opción "Diseñador de impresión". Se dibuja el área de impresión. Se introducen los datos del nuevo documento. Se presiona el botón "Guardar".
EC 2 Editar propiedades del documento con campos obligatorios vacíos.	El usuario intenta editar las propiedades del documento con campos obligatorios vacíos.	I vacío	V PDF	V A2	V 72 dpi	V Nuevo mapa de impresión	El sistema muestra el mensaje: "El campo título es de carácter obligatorio."	Se selecciona la opción "Diseñador de impresión". Se dibuja el área de impresión. Se introducen los datos del nuevo documento. Se presiona el botón "Guardar".
EC 3 Editar	El usuario intenta	I	V	V	V	V	Para todos los	Se selecciona la opción

propiedades del documento con datos incorrectos.	editar las propiedades del documento con datos incorrectos.	Mapa de Posicionamiento Global del Parque Jardín Botánico Nacional	PDF	A2	72 dpi	Nuevo mapa de impresión para el Jardín Botánico Nacional	casos el sistema mantiene la interfaz y muestra un mensaje de error. En el primer caso "El campo título admite hasta 50 caracteres". En el segundo caso "El campo nota de texto admite hasta 300 caracteres".	"Diseñador de impresión". Se dibuja el área de impresión. Se introducen los datos del nuevo documento. Se presiona el botón "Guardar".
		V	V	V	V	I		
		Mapa del Jardín Botánico Nacional	PDF	A2	72 dpi	Nuevo mapa de impresión para el Jardín Botánico Nacional. Contiene las especificaciones de las áreas verdes del parque, así como la información de las edificaciones de la misma...		

Serie Científica de la Universidad de las Ciencias Informáticas
Vol. 12, No.2, Mes Febrero, 2019
ISSN: 2306-2495 | RNPS: 2343
Pág. 1-15
http://publicaciones.uci.cu

En el proceso de aplicación de las pruebas se realizaron tres iteraciones utilizando el caso de prueba diseñado para detectar la mayor cantidad de errores en el funcionamiento del módulo.

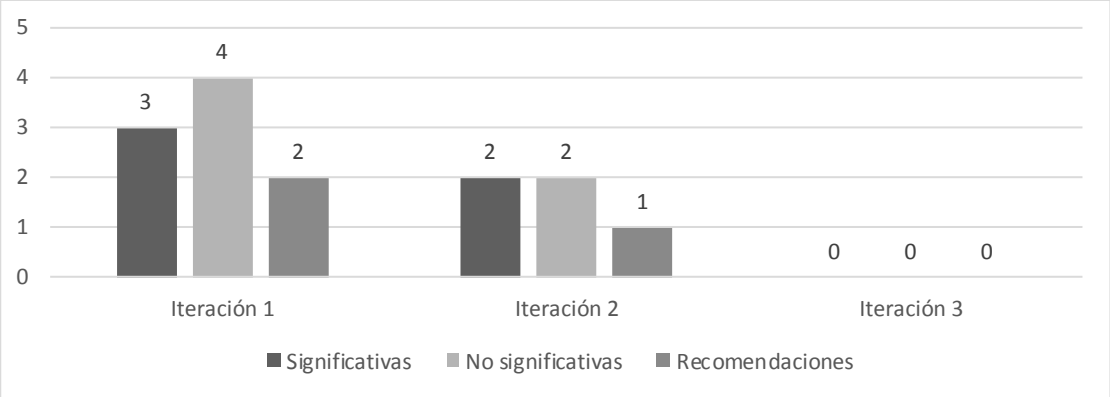

Fig. 6: Gráfico de no conformidades detectadas al realizar las pruebas.

En la Fig. 6 se evidencian las no conformidades detectadas al realizar las pruebas, estas fueron clasificadas atendiendo al nivel de afectación en el funcionamiento del módulo, en significativas, no significativas y recomendaciones. Al realizarse la primera iteración se detectaron 9 no conformidades de las cuales 3 fueron significativas al afectar el funcionamiento del módulo; como no significativas se detectó 1 error ortográfico, 3 errores de mensajes y las 2 restante fueron recomendaciones para mejorar el diseño del módulo. Una vez concluida la corrección de los errores detectados, se inicializó la segunda iteración donde se identificaron 5 no conformidades clasificadas en 2 significativa, 2 errores ortográficos en el caso de las no significativas y la restante fue 1 recomendación para solucionar los errores. Al terminar de solucionar los errores detectados se finalizó la segunda iteración y se dio inicio a una tercera iteración para continuar comprobando si existían errores. Al culminar la misma no se detectaron no conformidades.

Conclusiones

En el presente informe fueron abordados los temas referentes a la implementación de la propuesta de solución y la estrategia de pruebas a seguir durante la elaboración del sistema. La confección del diagrama de componente, permitió comprender la interacción entre los componentes que conforman el sistema y mostrar las dependencias que existen entre los mismos. Fueron definidos los estándares de codificación a emplear para la implementación de la aplicación, permitiendo la correcta uniformidad y organización del código. Se realizaron las pruebas pertinentes

(unitarias, integración, sistema), para garantizar la obtención de un producto con la calidad esperada. Según el criterio de expertos aplicado, el módulo es considerado como adecuado por el 95 % de los especialistas consultados. Las pruebas realizadas al módulo ayudaron a la identificación y corrección de 14 no conformidades, proporcionando mayor usabilidad y rendimiento a las funcionalidades de la solución; demostrando la satisfacción del cliente con los requisitos implementados.

Referencias

Leon, Johnny. 2016. [En línea] 14 de Noviembre de 2016. [Citado el: 13 de Febrero de 2019.] http://diagramasdecomponentes.blogspot.com/.

DELGADO, DAYANA H. BAILÓN. 2015. Lenguaje Unifido de Modeldo -UML. Calceta: s.n., 2015.

Moreno Barrios, José Antonio, y otros. 2015. Reglas de calidad para la codificación estandarizada en lenguaje C: una propuesta para la enseñanza a nivel superior. 2015. ISSN: 2395-9029.

Garrido, Alberto. 2017. QA: news. [En línea] 23 de Enero de 2017. [Citado el: 13 de Febrero de 2019.] https://qanewsblog.com/2017/01/23/que-contenido-se-debe-incluir-en-una-estrategia-de-pruebas-i/.

Quijano, Juan. 2018. Genbeta. [En línea] 8 de Octubre de 2018. [Citado el: 13 de Febrero de 2019.] https://www.genbeta.com/desarrollo/que-pruebas-debemos-hacerle-a-nuestro-software-y-para-que.

Panel Testing. 2015. [En línea] 11 de Febrero de 2015. [Citado el: 18 de Febrero de 2019.] https://www.panel.es/blog/software-qa-cuales-son-los-tipos-de-pruebas-software/.

Morales, Víctor Manuel Soto. 2018. [En línea] 2 de Mayo de 2018. [Citado el: 18 de Febrero de 2019.] https://www.pragma.com.co/blog/conoce-que-son-las-pruebas-no-funcionales-de-software.

Cantú, Andrea. 2017. [En línea] 4 de Septiembre de 2017. [Citado el: 19 de Febrero de 2019.] https://blog.acantu.com/que-son-pruebas-usabilidad/.

Terrera, Gustavo. 2017. [En línea] 26 de Febrero de 2017. [Citado el: 20 de Febrero de 2019.] https://testingbaires.com/2017/02/26/pruebas-caja-negra-enfoque-practico/.

Ruis Eguino, Sandra Berenice and Leon, Luis Vinicio. 2013. [Online] 22 Diciembre 2013. [Cited: 20 Febrero 2019.] https://sg.com.mx/revista/29/pruebas-usabilidad-web.

Prieto, Roexcy Vega, Rodríguez Luis, Zaylí and Justo Morell, Yaneisi Ofelia. 2013. Procedure for performing usability test. La Habana: s.n., 2013.

Rojas Rodríguez, María Elizabeth and Elgueta, Andrés Felipe Estay. 2017. Heurísticas de usabilidad/Experiencia del usuario en agencias virtuales de viaje. Valparaíso: s.n., 2017.

CRUZ MEDINA, Fanny Liliana; LÓPEZ DÍAZ, Andrea del Pilar; RUIZ CÁRDENAS, Consuelo. Sistema de Gestión ISO 9001-2015: Técnicas y herramientas de ingeniería de calidad para su implementación. 2017.