Informe técnico sobre análisis de código estático para el software de Contabilidad Versión 1.0

Perito: Mario Daniel Murcia Pinilla

Nombre: Mario Daniel Murcia Pinilla

Cedula: 1049659203

Email: mario.murcia@usantoto.edu.co

Contenido del informe

1. **Propósito de la evaluación**

Esta evaluación se realiza para comprobar el estado del software de contabilidad, su funcionamiento y rendimiento en multiplataforma, siguiendo los estándares de calidad del código fuente del software establecidos en ISO 25000 tales como comprobar la cantidad de código maloliente y densidad de comentario de este.

Bajo revisión del código identificar posibles errores o fallos que puedan llegar a afectar el software ya sea en su parte funcional o la documentación que debe tener para el entendimiento del código. Esto para que en próximas versiones sean tomadas en cuenta y mejoren la calidad que tiene el software en su versión actual.

2. Producto a evaluar

- Es un software de tipo empresarial realizado principalmente en php

- La versión actual del software es la 1.0
- Es un software para la gestión general de una empresa
- Posee una completa forma de facturación y múltiples medios para registros de los mismos.
- Mantiene registro de sus clientes.
- Mantiene registro de sus proveedores.
- Realiza cálculos de impuestos con iva, manteniendo todo legal.
- Tiene un apartado visual para facilitar el manejo del proyecto.

3. Especificación de la(s) herramientas usadas para medición

Sonarqube:este programa ayuda a evaluar el código fuente de proyectos programático, el programa está diseñado para ser de libre uso , por esta y por las muchas características que tiene para el análisis de código interactivo , se convierte en una de los mejores programas para hacer este trabajo.

Este programa nos ayuda a evaluar vulnerabilidades , bugs y código maloliente que pueda tener algún proyecto, dándonos una serie de posibles soluciones para el problema que tenga cada sección, ya sea un bug de desactualización de versiones o una mala redacción del código.

SonarQube está diseñado para funcionar con las herramientas más comunes y recomendadas de análisis de codigo por ejemplo Checkstyle y FindBugs.

Para el análisis de este proyecto nosotros utilizamos la versión ”8.3.1.34397” que corre a la vez con java “jdk-11.0.6+10” y este funciona en el puerto 9000 , esta versión corre sobre un contenedor de docker el cual tiene un tamaño de 1.7 gb.

SonarQube: Es uno de los plugins más utilizados en SonarQube que tiene como objetivo el análisis de código fuente, este plugin es de acceso libre y es el escaner por defecto de SonarQube.

La versión utilizada de Sonar Scanner corre sobre windows y es la 4.3.0.2102 .

Visual studio code: es un program que permite la visualización de archivos html, js , php, y cualquier archivo txt , Incluye soporte para la depuración, control integrado de Git, resaltado de sintaxis, finalización inteligente de código, fragmentos y refactorización de código. (Lardinois, Frederic (29 de abril de 2015).)

Este lo utilizamos para la visualización de el código fuente de el proyecto.

Docker: es un proyecto de código libre que nos permite virtualizar por medio de contenedores cualquier tipo de aplicaciones , ya sea desde un sistema operativo completo o un simple lector de texto , este fue utilizado para correr el servicio de SonarQube y poder hacer el análisis.

4. Especificación de la Norma de calidad

Una de las normas más importantes para la comunidad del desarrollo de software es la ISO 25000 , está norma se enfoca más en la Calidad de Software ya que es la unión de varias normas establecidas anteriormente.

Está norma se divide en 5 estandares fundamentales los cuales son.

- División para la medición de la calidad.
- División para la calidad del modelo.
- División para los requisitos de la calidad.
- División para la gestión de la calidad.
- División para la evaluación de la calidad.

Con el cumplimiento de esta misma se espera que el producto final tenga una mayor vigencia y permite que el proyecto tenga un estándar de calidad alto.

5. Requerimientos específicos de calidad funcionales y técnicos.

La empresa requiere que se evalué los siguientes requerimientos funcionales y técnicos específicos

1	Que el software pueda funcionar en sistemas operativos Android, MacOS, Windows XP, Windows 7 y Windows 10 (en 32 y 64 bits)	Alta
2	Que permita trabajar en forma rápida e intuitiva (cuente con ayudas visuales y auditivas interactivas en el software).	Media
3	Que tenga soporte multiidiomas, especialmente inglés y español	Media
4	Que permita adecuar su estilo de visualización para adecuarse a personas con limitaciones visuales (Ley 1680 de 20 de noviembre de 2013)	Baja
5	Implementación de Ley 1581 del 2012 – Protección de datos (HABEAS DATA)	Alta
6	Permita generar reportes en EXCEL Y PDF.	Media
7	Funcionalidad/modulo para reportar errores técnicos o funcionales desde el software.	Media
8	Permitir acceso a 100 usuarios simultáneos	Alta
9	Tolerancia a fallos (caída de red, apagones eléctricos frecuentes).	Media
10	Integración con office	Baja
11	Cumplimiento del 80% con el estándar OWASP, priorizando en las vulnerabilidades de robo de información, XSS, SQL injection y ransomware.	Alta
12	Capacidad de respaldo y recuperación de información desde el software.	Media
13	El software debe demandar mínimos recursos de hardware (cpu Intel celeron, 2 gigas de Ram)	Media

5.1.

1	Que el software pueda funcionar en sistemas operativos Android, MacOS, Windows XP, Windows 7 y Windows 10 (en 32 y 64 bits)	0%
2	Que permita trabajar en forma rápida e intuitiva (cuente con ayudas visuales y auditivas interactivas en el software).	0%
3	Que tenga soporte multiidiomas, especialmente inglés y español	100%
4	Que permita adecuar su estilo de visualización para adecuarse a personas con limitaciones visuales (Ley 1680 de 20 de noviembre de 2013)	0%
5	Implementación de Ley 1581 del 2012 – Protección de datos (HABEAS DATA)	0%
6	Permita generar reportes en EXCEL Y PDF.	50%
7	Funcionalidad/modulo para reportar errores técnicos o funcionales desde el software.	100%
8	Permitir acceso a 100 usuarios simultáneos	0%
9	Tolerancia a fallos (caída de red, apagones eléctricos frecuentes).	0%
10	Integración con office	0%
11	Cumplimiento del 80% con el estándar OWASP, priorizando en las vulnerabilidades de robo de información, XSS, SQL injection y ransomware.	0%
12	Capacidad de respaldo y recuperación de información desde el software.	0%
13	El software debe demandar mínimos recursos de hardware (cpu Intel celeron, 2 gigas de Ram)	100%

6. Atributos de calidad.

Los atributos de calidad que se utilizaran para la evaluación del Software Facturación, de acuerdo a lo especificado en el siguiente cuadrado:

Tabla 1 Atributos de calidad

ATRIBUTOS INTERNOS	Características del software que determinan su habilidad para satisfacer las necesidades propias e implícitas.
ATRIBUTOS EXTERNOS	Características del software que determinan su habilidad para satisfacer las necesidades explícitas e implícitas.
ATRIBUTOS EN USO	Características del software que determinan los requerimientos de los usuarios finales de manera que satisfagan sus necesidades.

6.1.Asignación de puntajes para atributos de calidad.

Los puntajes establecidos a los atributos de calidad seleccionados de acuerdo a las necesidades, se muestran en la siguiente tabla:

Tabla 2, Asignación de pesos sobre la medición de atributos.

Atributos internos	65
Atributos externos	35
Total	**100**

6.2.Atributos internos.

6.2.1. Tamaño de sistemas y código fuente (20%)

6.2.1.1. Lienas código:

Figura 1. Líneas de Código

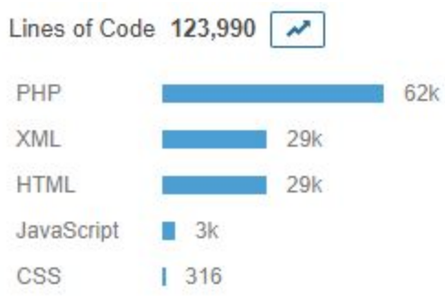

Figura 2. Líneas de Código-Lenguajes

Está valoracion se da gracias a SonarQube, herramienta que informa, que el proyecto está compuesto de 175.853 lineas de codigo de las cuales 62 mil están en código PHP , 29 mil están en código XML , 29 mil están en código HTML , 3 mil están en código JavaScript , 316 son CSS y las líneas restantes corresponden a líneas vacía.

6.2.1.2. Densidad De Código

Se calcula la densidad de código usando la fórmula que se muestra a continuación.

[Densidad de comentarios = líneas de código con comentario/(líneas de código+líneas de código con comentario)]

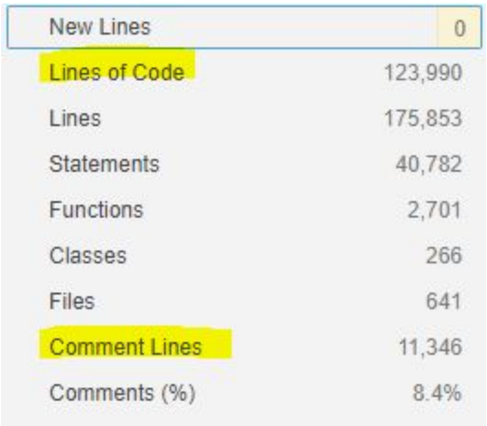

Figura 3. Líneas de Código-Líneas Comentadas

Como se observa en la imagen anterior , existen 123.990 líneas de código y 11.346 líneas comentadas , al aplicar la fórmula da como resultado que la densidad de código es igual a 0.0838 que se interpreta como 8.4%.

$$11,346/(123,990 + 11,346) = 0,0838$$

6.2.1.3. Duplicidad De código

Figura 4. Líneas Duplicadas

Como se muestra en la imagen la duplicidad de código del proyecto es de 27,781, lo que evidencia la mala estructuración del mismo.

6.2.2. Complejidad del software(20%)

6.2.2.1. Complejidad ciclomática

Figura 5. Complejidad ciclomática

Lo complejidad ciclomática en este caso es de 13,999 lo cual indica que el proyecto es no testeable y de muy alto riesgo.

6.2.2.2. Complejidad Cognitiva

Figura 6. Complejidad Cognitiva

Lo complejidad cognitiva en este caso es de 24,918 lo cual indica que el proyecto no es intuitivo y no es estructurado por tanto es muy difícil entenderlo.

6.2.3. Deuda técnica (20%)

6.2.3.1. Código Smell

Figura 7. Código Smell

Dentro del código del proyecto existe una gran cantidad de código smell, la cual es de 4,170 lo que indica deficiencias en el diseño las cuales pueden generar que se realice un desarrollo más lento y aumenta el riesgo de errores futuros.

6.2.3.2. Deuda Técnica

Figura 8. Deuda Técnica

Se puede observar la mala ejecución del código , al notar que la deuda técnica es de 94 días lo cual genera costos de tiempo y dinero innecesarios.

6.2.4. Seguridad(20%)

6.2.4.1. Vulnerabilidades

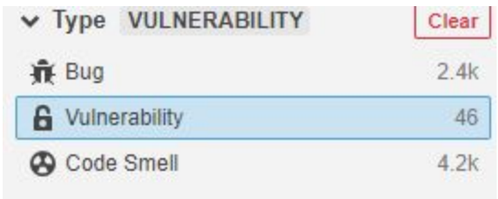

Figura 9. Vulnerabilidades

La gran cantidad de vulnerabilidades (46) y bugs que tiene el proyecto demuestra que la seguridad es muy baja por tanto no se recomienda la implementación de este proyecto.

6.2.4.2. OWASP

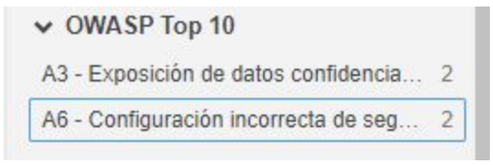

Figura 10. owasp

El proyecto intenta implementar certificados SSL/TLS propios en vez de utilizar los certificados recomendados por las Autoridad de certificación principal.

Posee una gran vulnerabilidad a la exposición de datos personales.

6.2.5. 3d code metrics (20%)

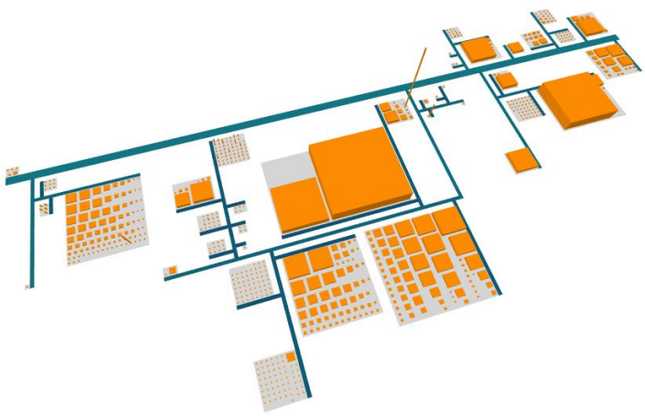

Figura 11. 3d code

6.2.5.1. Zonas en conflicto

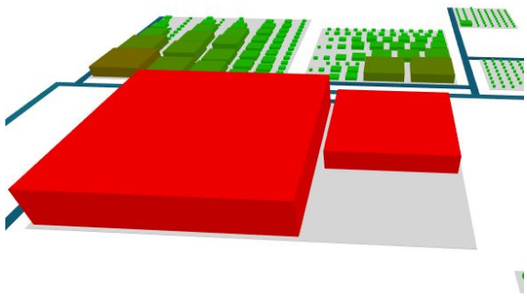

Figura 12. Zonas en conflicto

Con el color rojo se puede evidenciar la alta complejidad ciclomática.

6.2.5.2. Patrones

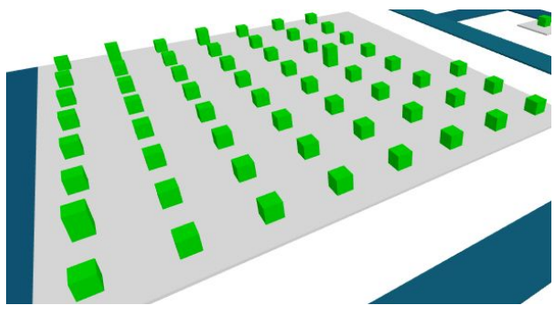

Figura 13. Patrones

Como se observa en la imagen existen sectores donde la complejidad ciclomática es baja y lo que se interpreta como un patrón.

6.2.5.3. Anomalías

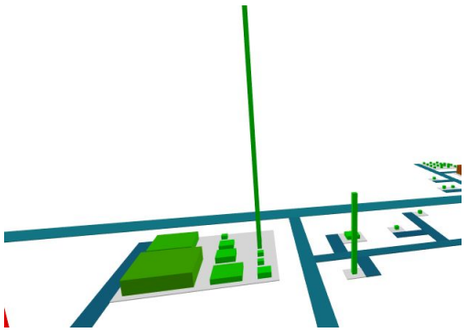

Figura 13. Anomalías

Como se observa en la imagen este sector tiene picos más altos , lo que significa que tiene muchas lineas de codigo pero poca complejidad. o cual se le denominan anomalías.

6.3. Atributos Externos.

6.3.1. Usabilidad (20%)

- La efectividad:ya que no se denota ninguna particularidad que demuestre que este proyecto está desarrollado en algún framework, y tampoco no se puede iniciar de una manera intuitiva es imposible determinar su efectividad , con el análisis del código podemos determinar que el proyecto si cumple con manejo de errores y con botones pero estos no están realizados de la mejor manera, no se especifica ningún cambio de color en ninguna opción, ya sea en botones o en textos de alertas.
  Evaluación correspondiente es igual a : malo
- la eficiencia: Ya que no ha sido posible la ejecución de este proyecto no se puede dar una valoración completa, pero con el análisis realizado al código ..... que no cumple con un sistema lógico de navegación, no existe coherencia en los nombres y en sus descripciones , y la evaluación de funcionalidad no se pudo llevar a cabo.
  Evaluación correspondiente es igual a : malo
- Satisfacción: Ya que no ha sido posible la ejecución de este proyecto no se puede dar una valoración completa, pero con el análisis realizado al código se puede decir que el proyecto no es cómodo para algún uso , el proyecto no cumple con la mayoría de requerimientos así que la aceptabilidad de este es baja.

  No se recomienda el uso de este proyecto puesto que no es intuitivo, no consta de ayudas auditivas para personas con discapacidades.

  Evaluación correspondiente es igual a : malo

6.3.2. Fiabilidad (30%)

El proyecto está diseñado de una forma muy mala y poco eficiente, puesto que no cumple con algún estándar para la organización del código, el proyecto no es confiable puesto que posee muchas vulnerabilidades y una de las más importantes es la exposición de datos personales.

La deuda técnica de este proyecto es de 94 días, lo cual hace prácticamente imposible la corrección del mismo, no se recomienda la corrección del proyecto pues esto llevaría mucho tiempo y muchos costos económicos para la empresa.

El proyecto no cuenta con malware/virus pero sus inmensas vulnerabilidades crean brechas de seguridad y esto lo hace muy vulnerable a hacker maliciosos que añaden estos malware/virus a el proyecto.

Por lo mencionado anteriormente el proyecto no es fiable para el manejo o uso.

6.3.3. Mantenibilidad (50%)

Al tener una deuda técnica de 94 días, es muy difícil corregir los fallos que tiene. Corregir esto generaría un inmenso gasto de recursos tanto personal como económico, lo cual no genera un beneficio al corregir estos errores.

7. Recomendaciones y conclusiones

7.1. Conclusión

7.1.1. Seguridad

El proyecto no cuenta con protocolos de seguridad eficientes que puedan mantener protegidos archivos fundamentales y de un índole privado.

7.1.2. Confiabilidad

Por las inmensas vulnerabilidades que presenta no es confiable su uso en la versión actual.

7.1.3. Usabilidad

Es imposible el uso de este proyecto en un ámbito profesional.

7.2. Recomendaciones

- No usar este proyecto en su estado actual.
- No invertir recursos en su mejoramiento.

8. Bibliografía

Citación 1 : Lardinois, Frederic (29 de abril de 2015). Visual Studio Code. Recuperado de https://techcrunch.com/2015/04/29/microsoft-shocks-the-world-with-visual-studio-code-a-free-code-editor-for-os-x-linux-and-windows.

Figura 1. Líneas de Código ----- https://www.sonarqube.org/

Figura 2. Líneas de Código-Lenguajes ----- https://www.sonarqube.org/

Figura 3. Líneas de Código-Líneas Comentadas ----- https://www.sonarqube.org/

Figura 4. Líneas Duplicadas ----- https://www.sonarqube.org/

Figura 5. Complejidad ciclomática ----- https://www.sonarqube.org/

Figura 6. Complejidad Cognitiva ----- https://www.sonarqube.org/

Figura 7. Código Smell ----- https://www.sonarqube.org/

Figura 8. Deuda Técnica ----- https://www.sonarqube.org/

Figura 9. Vulnerabilidades ----- https://www.sonarqube.org/

Figura 10. owasp ----- https://www.sonarqube.org/

Figura 11. 3d code ----- https://www.sonarqube.org/

Figura 12. Zonas en conflicto ----- https://www.sonarqube.org/

Figura 13. Patrones ----- https://www.sonarqube.org/

Figura 13. Anomalías

9. Firma del perito.

Responsable de la evaluación **Firma**

Nombre: Mario Daniel Murcia Pinilla
Empresa: TSK STUDIOS
Cargo: Arquitecto y tester
GitHub: mariomurcia

Mario Daniel Murcia Pinilla