DOCUMENTO DE ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE

Equipo 2

9 de mayo de 2025

Realizó Mantenimiento:

Canul Ordoñez, Josué Israel

Garcilazo Cuevas, Mónica

Leo Fernández, José Carlos

Pool Flores, Endrick Alfredo

Rodríguez Coral, Samuel David

1. Control de documentación

1.2. Control de configuración

Título:	Documento de especificación de requerimientos de software
Referencia:	N/A
Autor:	Dioné Martín
Fecha:	19 de febrero de 2025

1.3. Histórico de versiones

Versión	Fecha	Estado	Responsable	Nombre del archivo
1.0.1	19 – 02 – 2025	A	Dioné Martín	Documento_ERS_1.0.1
1.0.2	24 – 02 – 2025	A	Adjany Armenta	Documento_ERS_1.0.2
1.0.3	26 – 02 – 2025	A	Dioné Martín	Documento_ERS_1.0.3
1.0.4	27 – 02 – 2025	A	Adjany Armenta	Documento_ERS_1.0.4
2.0.0	01 – 04 - 2025	B	Arturo Quezada Daniel Rosado	Documento_ERS_2.0.1
3.0.0	09 – 05 – 25	A	Endrick Pool	Documento_ERS_3.0.0

Estado: (B)orrador, (R)evisión, (A)probado

1.4. Histórico de cambios

Versión	Fecha	Cambios
1.0.1	19 – 02 – 2025	Se creó la estructura del documento y se completaron las secciones Propósito, Alcance, Requisitos funcionales, Requisitos no funcionales y Referencias.
1.0.2	24 – 02 – 2025	Se modificaron los requisitos no funcionales.
1.0.3	26 – 02 – 2025	Se modificó la estructura de formato y redacción de los requisitos específicos.

1.0.4	27 – 02 – 2025	Se modificó el formato con base al estándar de documentación.
2.0.0	01 – 04 – 2025	Se modificó el alcance para adaptarlo a los nuevos requisitos. Se agregaron los requisitos para hacer el conteo de métodos por clase (2.3), se modificó el requisito 2.2.1 para incluir tanto el conteo por cada clase como el de todas las clases. Se modificó el requisito de impresión (3.1) y se agregó un nuevo requisito de impresión (3.2). Se eliminaron los requisitos relacionados con el conteo de líneas lógicas.
3.0.0	09 – 05 – 2025	Se modificó el alcance del proyecto y se agregaron los requisitos funcionales relacionados con la nueva característica de la comparación de proyectos que se desea implementar.

Contenido

2. Introducción

2.1. Propósito

El siguiente documento tiene como propósito guiar el desarrollo del Sistema de conteo de Líneas de Código (LOC) en Java, asegurando que cumpla con los estándares establecidos por el equipo en la materia de Mantenimiento de Software, para su correcto funcionamiento y verificación.

2.2. Alcance

El producto por elaborar es un sistema en Java que analiza archivos .java para contar el total de métodos dentro de cada clase del programa, el total de líneas físicas por clase, y el total de líneas físicas del programa. Excluyendo, para el conteo de líneas físicas, los comentarios y líneas en blanco.

Además, el sistema permitirá comparar dos versiones distintas de un proyecto de software. Ejecutando el análisis anterior para cada proyecto y posteriormente comparación de los proyectos clasificando los cambios en líneas modificadas, eliminadas y nuevas. Como adición a esta función, el sistema generará reportes individuales por clase con los resultados de la comparación.

2.3. Objetivos

Proporcionar una herramienta precisa para medir el código fuente conforme a los estándares de conteo y codificación definidos por el equipo, con el fin de facilitar su verificación y mantenimiento. Además, ofrecer una opción sencilla para comparar dos versiones de un proyecto de software, permitiendo un análisis efectivo de los cambios realizados.

2.4. Referencias

- Estándar de codificación definido por el equipo.
- Estándar de conteo para las líneas de código definido por el equipo.

3. Requerimientos específicos

3.1. Requerimientos Funcionales (RF)

RF- 01. Ingreso de archivos de código fuente

1.1. Acceso mediante terminal

1.1.1. El sistema debe permitir al usuario ingresar manualmente desde la terminal o consola de comandos la ruta de un directorio que contenga los archivos de código fuente.

1.1.2. El sistema debe permitir al usuario ingresar manualmente desde la terminal o consola de comandos la ruta de dos directorios que contenga los archivos de código fuente de dos proyectos diferentes para comparar.

1.2. Validación de la ruta del proyecto

1.2.1. El sistema debe verificar que la ruta ingresada por el usuario corresponda a un directorio válido en el sistema de archivos.

1.2.2. Si la ruta ingresada no es válida, el sistema no seguirá con el proceso.

1.2.3. Si el sistema no encuentra archivos .java, no seguirá con el proceso.

RF- 02. Conteo de líneas de código (LOC)

2.1. Identificación de archivos para procesamiento

2.1.1. El sistema debe identificar y procesar únicamente archivos con extensión .java.

2.1.2. El sistema debe recopilar recursivamente todos los archivos con extensión .java que se encuentran dentro del directorio especificado y sus subdirectorios.

2.2 Conteo de líneas físicas (LOC Físicas)

2.2.1. El sistema debe identificar y contar correctamente las líneas físicas de acuerdo con el estándar de conteo previamente definido. Específicamente:

2.2.1.1. El sistema debe contar la cantidad de líneas físicas por cada clase.

2.2.1.2. El sistema debe contar el total de líneas físicas de todas las clases del programa.

2.2.2. El sistema no debe contar líneas en blanco, líneas de comentarios, tabulaciones o espacios.

2.2.3. Si el archivo .java no contiene ninguna clase, este se ignorará para el conteo de líneas físicas de código.

2.3. Conteo de métodos

2.3.1. Si el archivo .java no contiene ninguna clase, este se ignorará para el conteo de métodos por clase.

2.3.2. El sistema deberá contar la cantidad de métodos dentro de cada clase.

2.3.3. El sistema debe identificar y contar la cantidad de métodos de acuerdo con el estándar de conteo previamente definido.

2.3.4. Si la clase no contiene ningún método, deberá retornar el valor 0 como el número total de métodos.

2.3.5. El sistema debe almacenar en memoria el total de métodos por clase para su presentación posterior.

RF- 03. Generación y presentación de resultados

3.1. El sistema debe presentar en consola un resumen estructurado en el cual, por cada clase del programa, se indicará la siguiente información. Los valores A y B coinciden con los calculados en RF 2.3.2 y RF 2.2.1.1, respectivamente*:*

Nombre de la clase:

Total de Métodos: A

LOC Físicas por Clase: B

3.2. El sistema debe presentar en consola al final el total de líneas físicas del programa. El valor C coincide con el valor calculado en RF 2.2.1.2:

Total de líneas físicas de código: C

RF- 04. Validación de código y estructura

4.1. Cumplimiento del estándar de codificación

4.1.1. El sistema debe validar que los archivos .java cumplan con el estándar de codificación establecido previamente.

4.1.2. El sistema debe asegurarse de que el nombre de los archivos coincida con el nombre de la clase de nivel superior, respetando mayúsculas y minúsculas.

4.1.3. El sistema debe verificar que no existan importaciones con comodines (*).

4.1.4. El sistema debe validar el formato de indentación con dos espacios por nivel.

4.2. Separación de responsabilidades

4.2.1. Cada uno de los módulos que compone al sistema debe ser independiente.

RF- 05. Comparación de dos proyectos de software

5.1 El sistema debe permitir al usuario ingresar manualmente, desde la terminal o consola de comandos, la ruta de dos directorios correspondientes a diferentes versiones de un proyecto de software.

5.2 El sistema debe realizar el conteo de líneas de código en ambos proyectos, tal cual se específica en RF 2 y RF 3. Para posteriormente ejecutar un proceso de comparación entre ellos.

5.2 El sistema debe identificar y clasificar las diferencias encontradas entre los proyectos en 5 categorías:

- Líneas originales
- Líneas modificadas
- Líneas eliminadas
- Líneas nuevas

- Líneas cortadas

RF- 06. Generación de reportes de comparación

6.1 El sistema debe generar reportes individuales por clase que contengan los cambios realizados entre dos versiones del proyecto.

6.2 El sistema debe permitir al usuario ingresar manualmente desde la terminal o consola de comandos la ruta de un directorio para guardar los reportes de comparación.

6.3. Validación de la ruta de guardado de los reportes

6.2.1. El sistema debe verificar que la ruta ingresada por el usuario corresponda a un directorio válido en el sistema de archivos.

6.2.2. Si la ruta ingresada no es válida, el sistema no seguirá con el proceso.

3.2. Requerimientos No Funcionales (RNF)

RNF - 01. Mantenibilidad y modularidad

1.1. El código del sistema debe contener comentarios explicativos en cada función clave para facilitar futuras modificaciones.

1.2. La arquitectura del sistema debe seguir un diseño modular.

1.3. Los módulos deben permitir modificaciones sin afectar el funcionamiento del sistema en su totalidad.

RNF - 02. Fiabilidad y precisión

2.1. El sistema debe garantizar la precisión en el conteo de líneas de código de acuerdo con el estándar de conteo establecido.