

Facultad de matemáticas

Especificación de Requisitos de Software

Versión 1.0

Autores:

Aaron Isaac Graniel Arzat

Fernando Joachín Prieto

David Peña Muñoz

Jose Luis Pooc Moo

Andrea Isabel Torres Perez

**CONTROL DE DOCUMENTACIÓN**

**Control de Configuración**

| **Título:** | Especificación de Requisitos de Software |
| --- | --- |
| **Referencia:** | ESP\_EspecificaciónRequisitosSoftware\_v1.0\_2025-02-24 |
| **Autor(es):** | Aaron Isaac Graniel Arzat |
| **Fecha:** | 24/02/2025 |

**Histórico de versiones**

| **Versión** | **Fecha** | **Estado** | **Responsable** | **Nombre de archivo** |
| --- | --- | --- | --- | --- |
| 1.0 | 24/02/2025 | A | Aaron Isaac Graniel Arzat | ESP\_EspecificaciónRequisitosSoftware\_v1.0\_2025-02-24 |

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

**Histórico de cambios**

| **Versión** | **Fecha** | **Cambios** |
| --- | --- | --- |
| 1.0 | 24/02/2025 | Ninguna, primera versión. |
| 1.0 | 26/02/2025 | Se ajustó la definición de líneas lógicas para alinearse con los criterios establecidos. |

**Índice**

[**1. Introducción 4**](#_9wjat9p471x)

[1.1 Propósito 4](#_1yo56vo41nfe)

[1.2 Alcance 4](#_gvkp0ugg02a4)

[1.3 Definiciones, Acrónimos y Abreviaturas 4](#_128prw2ai5hq)

[1.4 Referencias 5](#_lrbfgfxohutt)

[**2. Descripción General 5**](#_e1j80l4i7683)

[2.1 Perspectiva del Producto 5](#_rdt0bjupnkd6)

[2.2 Funciones del Producto 5](#_tvu75nmm37a1)

[2.3 Suposiciones y Dependencias 6](#_x0ztnynztvpt)

[**3. Requisitos Específicos 7**](#_71eb5vp8wzqc)

[3.1 Requisitos Funcionales 7](#_c4syg1c49ual)

[RF-01: Gestión de directorios 7](#_dkee142xhuhf)

[RF-02: Validación del Archivo 7](#_zbrlrky5iclg)

[RF-03: Gestión de archivos 7](#_z19q4p9n9j06)

[RF-04: Eliminación de Comentarios y Líneas Vacías 7](#_lyqtpe9831zj)

[RF-05: Conteo de Líneas Lógicas 8](#_1vqizpotc84b)

[RF-06: Conteo de Líneas Físicas 8](#_kpf0imxiys6p)

[RF-07: Presentación de Resultados 8](#_n01xh16rrh4z)

[3.2 Requisitos de Interfaz Externa 8](#_pm5arwc31gg9)

[Interfaz de Usuario 8](#_n6ooxybaqypa)

[Interfaz de Archivo 8](#_o6te0cukuyff)

[**4. Anexos 9**](#_983r8x7nnqrj)

[4.1 Diagrama de clases 9](#_65k2wn7if4ak)

# **1. Introducción**

## **1.1 Propósito**

El presente documento tiene como objetivo definir de manera exhaustiva los requisitos necesarios para desarrollar una herramienta de análisis de código fuente en Java. Esta aplicación está diseñada para automatizar el procesamiento de archivos de código, permitiendo la determinación precisa de dos métricas esenciales: el número de líneas lógicas y el número de líneas físicas.

## **1.2 Alcance**

El alcance del sistema abarca el análisis y procesamiento de archivos fuente escritos en lenguaje Java. Concretamente, el software se encargará de:

* **Verificar la Validez del Archivo:** Asegurar que el archivo a analizar posee la extensión ".java", lo que garantiza que se encuentra en un formato compatible y esperado por la herramienta.
* **Conteo de Líneas:** Aplicar una serie de reglas particulares establecidas en el Estándar de Conteo para dos tipos de líneas:
  + **Lógicas:** Son aquellas líneas que establecen la estructura y organización del programa, definiendo su arquitectura básica y flujo de control. Incluyen elementos clave como declaraciones de clases, métodos y estructuras que guían la lógica del código.
  + **Físicas:** Cada línea que, tras la eliminación de comentarios y espacios en blanco, representa una porción efectiva de código.
* **Presentación de Resultados:** Los resultados del análisis se presentarán en la consola mediante un formato tabulado, lo que facilitará su lectura e interpretación por parte del usuario.

## **1.3 Definiciones, Acrónimos y Abreviaturas**

Para evitar ambigüedades y asegurar que todos los lectores del documento tengan un entendimiento común de los términos utilizados, se definen a continuación algunos conceptos clave:

* **LOC:** Abreviatura de “Líneas de Código”, utilizada para referirse al conteo de líneas en un programa.
* **LOC Lógicas:** Se refiere al conteo de líneas que definen la arquitectura y flujo del código, como elementos estructurales clave. Excluyen detalles operativos o instrucciones simples.
* **LOC Físicas:** Se refiere al conteo de todas las líneas de código efectivas después de eliminar comentarios y líneas vacías, proporcionando una medida directa de la extensión del código fuente.
* **ERS:** Especificación de Requisitos de Software, documento que recoge de manera sistemática las necesidades y funcionalidades que debe cubrir el sistema.

## **1.4 Referencias**

* **IEEE Std 830-1998:** "Recommended Practice for Software Requirements Specifications", que establece las directrices y estructura para la documentación de requisitos de software.
* **Estándar de Conteo de Líneas de código físicas y lógicas.** Documento interno que define los criterios y reglas para el conteo de líneas lógicas y físicas en código fuente.

# 

# **2. Descripción General**

## **2.1 Perspectiva del Producto**

El Contador de Líneas de Código se concibe como una herramienta sencilla y eficiente destinada a obtener datos cuantitativos sobre la extensión de los programas escritos en Java. Su función principal es contar, de forma automatizada, tanto las líneas lógicas como las físicas del código, ofreciendo información precisa y estructurada que facilita la medición de la cantidad de código presente en un archivo.

## **2.2 Funciones del Producto**

El sistema incorpora una serie de funcionalidades que se describen a continuación de manera detallada:

* **Gestión de Directorio:**

El sistema cuenta con un módulo encargado de gestionar directorios. Este módulo verifica que el directorio especificado exista y, si no es así, notifica al usuario con un mensaje de error. Una vez validado el directorio, el sistema explora su contenido y extrae todos los archivos con extensión .java para su análisis. Si durante la exploración del directorio surgen problemas, como falta de permisos, el sistema informará al usuario de manera clara y precisa.

* **Validación de Archivo:**Antes de iniciar cualquier proceso de análisis, el sistema verifica que el archivo de entrada cumpla con ciertos aspectos descritos en el estándar de codificación. Este paso es fundamental para asegurar que el archivo cumpla con el formato esperado y evitar el procesamiento de archivos que no pertenezcan al dominio de la aplicación.
* **Gestión de Archivos**

Además, el sistema incluye un módulo dedicado a la gestión de archivos, el cual se encarga de extraer todas las líneas del archivo para su posterior procesamiento, almacenándolas en una estructura de datos adecuada. En caso de que ocurran errores durante la lectura, como inexistencia del archivo, el sistema notificará al usuario con mensajes descriptivos.

* **Conteo de Líneas Lógicas:**El sistema implementa una lógica específica para identificar y contar las líneas que definen la estructura y flujo del programa, de acuerdo con lo establecido en el Estándar de Conteo. Este estándar prioriza elementos clave como declaraciones de clases, métodos y estructuras de control (if, for, while, switch, try/catch), excluyendo instrucciones simples, declaraciones de variables, comentarios y líneas de cierre de bloques. De esta manera, se garantiza que la métrica refleje con precisión la arquitectura y organización del código, no su implementación operativa.
* **Conteo de Líneas Físicas:**Además del conteo lógico, se realiza un conteo de todas las líneas efectivas del código, es decir, aquellas que aportan a la ejecución del programa. Conforme al Estándar de Conteo de Líneas (ver Referencias), antes de proceder al conteo se eliminarán los comentarios y las líneas en blanco, lo que permite obtener una estimación directa y precisa del tamaño real del código fuente.
* **Presentación de Resultados:**Finalmente, el sistema organiza y presenta los resultados obtenidos en un formato tabulado en la consola. Este formato facilita la lectura y comparación de los datos, permitiendo a los usuarios identificar rápidamente el número de líneas lógicas y físicas correspondientes a cada archivo analizado.

## **2.3 Suposiciones y Dependencias**

Para el correcto funcionamiento del sistema se han establecido las siguientes suposiciones y dependencias:

* **Entrada Válida:** Se asume que el usuario proporcionará una ruta y nombre de archivo correctos, y que dicho archivo será un código fuente Java válido.
* **Formato Estándar:** El sistema depende de que el código fuente se adhiera a la sintaxis y convenciones del lenguaje Java, lo que permite aplicar correctamente las reglas de eliminación de comentarios y conteo de líneas.
* **Recursos del Sistema:** Se supone que el entorno de ejecución dispone de una JVM compatible.

# **3. Requisitos Específicos**

## **3.1 Requisitos Funcionales**

El sistema debe cumplir con una serie de requisitos funcionales que garantizan su correcto desempeño y aseguran la coherencia en el procesamiento de los archivos de código.

### **RF-01: Gestión de directorios**

El sistema extrae todos los archivos .java dentro de él y procesa el conteo de líneas físicas y lógicas para cada archivo. Si el directorio no existe o hay errores de acceso, se mostrará un mensaje de error.

### **RF-02: Validación del Archivo**

El sistema deberá comprobar que el archivo proporcionado cumpla con las siguientes reglas de formato:

* **Longitud de Línea:** Asegurar que cada línea del archivo no exceda los 120 caracteres.
* **Estilo de Llaves:** Comprobar que, en las líneas que finalizan con una llave de apertura “{“, ésta esté precedida por una declaración válida (por ejemplo, una definición de clase, método o estructura de control).
* **Instrucciones Ejecutables Múltiples:** Validar que cada línea contenga, como máximo, una instrucción ejecutable, salvo en los casos excepcionales permitidos (como en la estructura de un bucle for).

En caso de que alguna de estas condiciones no se cumpla, el sistema mostrará un mensaje de error específico, indicando la naturaleza del problema.

### 

### **RF-03: Gestión de archivos**

El sistema extrae las líneas de los archivos javas para su procesamiento. Si el archivo no existe o hay errores de lectura, se mostrará un mensaje de error.

### 

### **RF-04: Eliminación de Comentarios y Líneas Vacías**

El programa debe procesar el contenido del archivo de la siguiente manera:

* **Eliminar comentarios de línea:** Descartar todas las líneas que comiencen con //.
* **Eliminar bloques de comentarios:** Remover los segmentos de código delimitados por /\* y \*/.
* **Descartar líneas vacías:** Eliminar aquellas líneas que, tras el procesamiento, no contengan código efectivo.

### 

### **RF-05: Conteo de Líneas Lógicas**

El sistema debe implementar una funcionalidad para calcular el número de líneas lógicas en el código fuente. Se deben aplicar reglas específicas de acuerdo con el estándar de conteo establecido para este tipo de métrica.

### **RF-06: Conteo de Líneas Físicas**

El programa debe contar todas las líneas físicas presentes en el archivo una vez eliminados los comentarios y líneas vacías. Cada línea de código que permanezca después del procesamiento será considerada en este conteo.

### **RF-07: Presentación de Resultados**

El sistema debe mostrar los resultados del conteo de manera estructurada y comprensible. La salida incluirá el nombre del archivo analizado, la cantidad de líneas físicas y la cantidad de líneas lógicas identificadas. Esta información será presentada en la consola mediante un formato tabular.

## 

## **3.2 Requisitos de Interfaz Externa**

El sistema interactúa con el usuario y con los archivos a través de interfaces diseñadas para garantizar una experiencia funcional eficiente y sin complicaciones.

### **Interfaz de Usuario**

La herramienta se ejecuta en un entorno de línea de comandos, donde el usuario proporciona la ruta del archivo a analizar. La salida de los resultados se mostrará en la consola, utilizando un formato estructurado que destaque cada una de las métricas obtenidas. Se espera que la interfaz sea intuitiva y que incluya mensajes informativos o de error que orienten al usuario durante su uso.

### **Interfaz de Archivo**

El sistema debe ser capaz de leer archivos de texto que contengan código fuente en Java desde un directorio especificado. Se garantizará que el proceso de lectura sea robusto, manejando adecuadamente situaciones como archivos inexistentes o rutas inválidas.

# 

# **4. Anexos**

## **4.1 Diagrama de clases**

El diagrama de clases incluido en este anexo ilustra la estructura fundamental del sistema "Contador de Líneas de Código". En él se representan las clases principales, como FileFormatValidator, encargada de validar el formato de los archivos; FileManager y DirectoryManager, responsable de la lectura y manejo de archivos y directorios; y la jerarquía de LineCounter, compuesta por la clase abstracta y sus implementaciones concretas, LogicalLineCounter y PhysicalLineCounter, que se encargan de contar las líneas lógicas y físicas respectivamente.

*Imagen 1 - Diagrama de clases*