PLAN DE ADMINISTRACIÓN DE LA CONFIGURACIÓN

Equipo 2

18 de febrero 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

1.1 Control de configuración

Título:	Estándar de conteo
Referencia:	N/A
Autor:	Cristian David Pan Zaldivar
Fecha:	9 de febrero de 2025

1.2 Histórico de versiones

Versión	Fecha	Estado	Responsable	Nombre del archivo
1.0.1	16 – 02 – 2025	A	Cristian Pan	Plan_administracion_de_ configuración_1.0.1
2.0.0	05 – 05 – 2025	A	José Carlos Leo Fernández	Plan_administracion_de_ configuración_2.0.0

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

1.3 Histórico de cambios

Versión	Fecha	Cambios
1.0.1	09 – 02 – 2025	Se creó la estructura, se añadieron las descripciones.
2.0.0	05 – 05 – 2025	Se modificó el nombramiento de los documentos.

CONTENIDO

1 Control de documentación 1
1.1 Control de configuración 1
1.2 Histórico de versiones 1
1.3 Histórico de cambios 1
2 Introducción 3
2.1 Propósito 3
2.2 Alcance 3
2.3 Cronograma 3
3 Herramientas de administración de la configuración 3
4 Información documentada relacionada 4
5 Políticas 4
5.1 Políticas sobre la práctica de la administración de la configuración 4
6 Calificación y capacitación 4
7 Criterios para la selección de elementos de configuración 4
7.1 Terminología 5
8 Identificación de la configuración 5
8.1 Estructura de desglose de elementos de configuración 5
8.2 Convenciones de nombres y números 6
8.2.1 Ramas (branches) 6
8.2.2 Commits: 6
8.2.3 Para los documentos: 6
9 Método para identificar el estado de revisión 7
9.1 Para el código fuente: 7
9.2 Para los documentos: 7
10 Líneas base de configuración 7
11 Control de cambios 8
11.1 Documentación 8
11.1.1 Control de configuración 8
11.1.2 Histórico de versiones 8
11.1.3 Histórico de cambios 8
11.2 Código fuente: 8
12 Métodos para procesar cambios y concesiones: 9
12.1 Para los documentos: 9
12.2 Para el código fuente: 9

2 Introducción

2.1 Propósito

Este Plan de Administración de la Configuración (PAC) tiene como propósito definir y gestionar los elementos de configuración relacionados con el desarrollo del programa solicitado en la materia de Mantenimiento de Software, asegurando que tanto el código fuente como los documentos asociados sean controlados, versionados y actualizados de manera estructurada usando como referencia el *ISO 10007 Quality management — Guidelines for configuration management* (Anexo A).

2.2 Alcance

Este PAC abarca los siguientes elementos:

- Código fuente desarrollado en Java, encargado de:
- Documentos de soporte:
    - Especificación de Requisitos de Software (ERS).
    - Documento de Diseño de Arquitectura.
    - Documento de Casos de Pruebas.
    - Manual de Usuario.

2.3 Cronograma

El proyecto es un programa en Java, dividido en tres entregas, con funcionalidades específicas en cada fase:

- Primera entrega (27 de febrero)
- Segunda entrega (27 de marzo)
- Tercera entrega (16 de mayo)

3 Herramientas de administración de la configuración

- Git y GitHub: Para el control de versiones y gestión de ramas.
    - Conventional Commits: Para estandarizar los mensajes de los commits.
    - Branches: Las ramas permiten desarrollar tareas específicas de forma aislada, sin interferir con el resto del repositorio.
- Java: Lenguaje de programación utilizado.

4 Información documentada relacionada

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

5 Políticas

5.1 Políticas sobre la práctica de la administración de la configuración

Las políticas de administración de la configuración establecen que todo el código debe seguir los estándares de codificación y conteo definidos por el equipo. Los cambios en el código deben ser propuestos mediante *pull requests* en GitHub cada *pull request* debe estar ligado a un issue de la lista de issues que se encuentra en GitHub, los cuales deben ser revisados y aprobados por al menos una persona antes de ser fusionados con la rama principal. Los mensajes de los *commits* deben seguir el formato de Conventional Commits para garantizar claridad y consistencia. Cada rama creada corresponde a una funcionalidad nueva, corrección o mejora específica.

Para cada documento generado, que es parte de los elementos de la configuración, en la primera página después de la portada se llevará un control de documentación.

6 Calificación y capacitación

Todos los miembros del equipo deben estar familiarizados con Git, GitHub, Java, el estándar de codificación y el estándar de conteo.

7 Criterios para la selección de elementos de configuración

Los elementos de configuración se seleccionan con base en los siguientes criterios:

- Relevancia para el funcionamiento del software: Se incluyen los archivos del código fuente (.java) porque constituyen la base funcional del programa desarrollado en Java, y cualquier cambio en ellos afecta directamente el comportamiento del sistema.
- Guías para la consistencia y calidad: El Documento de Estándar de Codificación y el Documento de Estándar de Conteo se seleccionan porque establecen las reglas que el equipo debe seguir para garantizar uniformidad en el desarrollo y medición del código, siendo esenciales para la calidad del producto.

- Definición y validación del sistema: El Especificación de Requisitos de Software (ERS) se incluye debido a que define los requisitos funcionales y no funcionales que guían todo el desarrollo, mientras que el Documento de Casos de Prueba asegura que el software cumpla con esos requisitos mediante pruebas estructuradas.
- Estructura y diseño técnico: El Documento de Arquitectura se selecciona porque describe los componentes principales del sistema y sus interacciones, siendo crítico para mantener la integridad del diseño durante el desarrollo.
- Soporte al usuario final: El Manual de Usuario se incorpora como elemento de configuración ya que proporciona las instrucciones necesarias para el uso del programa, siendo un entregable clave para los usuarios.

7.1 Terminología

- Branch: Rama en el repositorio que representa un desarrollo aislado.
- Issue: Tarea o problema registrado en GitHub para seguimiento.
- Pull request: Solicitud para integrar cambios de una rama a otra tras revisión.

8 Identificación de la configuración

8.1 Estructura de desglose de elementos de configuración

- Código fuente: Programas en Java para análisis de líneas y comparación de versiones.
- Documentos relacionados:
    - ERS: Define los requisitos funcionales y no funcionales del programa.
    - Diseño de Arquitectura: Describe los componentes principales del sistema y sus interacciones.
    - Casos de Pruebas: Contienen los casos específicos para validar el cumplimiento de los requisitos.
    - Manual de Usuario: Proporciona instrucciones para usar el programa desarrollado.
    - Documento de arquitectura: Describe los componentes principales del sistema y sus interacciones
    - Estándar de codificación: Describe el estándar a seguir para escribir el código
    - Estándar de conteo: Describe la manera en la que se cuentan las líneas lógicas y físicas
    - Estimación de tamaño: Calculo de tamaño del programa

8.2 Convenciones de nombres y números

Las siguientes convenciones se aplicarán para garantizar consistencia y trazabilidad en el proyecto:

8.2.1 Ramas (branches)

Todas las ramas seguirán el formato [tipo]/[acción], donde:

[tipo] indica el propósito de la rama:

feature: Para nuevas funcionalidades (ejemplo: feature/add-line-counter).

bugfix: Para correcciones de errores (ejemplo: bugfix/fix-crash-on-startup).

hotfix: Para correcciones urgentes en producción (ejemplo: hotfix/patch-security-issue).

docs: Para actualizaciones en documentación técnica (ejemplo: docs/update-readme).

[acción] describe brevemente la tarea (en minúsculas, con guiones para separar palabras).

Ejemplo completo: feature/implement-user-login.

8.2.2 Commits:

Los mensajes de commit seguirán el estándar Conventional Commits con el formato [tipo]: [descripción], donde:

[tipo] puede ser:

feat: Nueva funcionalidad.

fix: Corrección de un error.

docs: Cambios en documentación.

refactor: Reorganización del código sin cambiar funcionalidad.

[descripción] es una explicación breve y específica del cambio (en minúsculas).

Ejemplos: feat: add line counting function, fix: correct null pointer exception.

8.2.3 Para los documentos:

8.2.3.1 Nombres de archivo:

Los documentos llevarán un nombre que refleje su contenido y versión, en el formato [nombre]_[nombre]_X.Y.Z.[extensión], donde:

[nombre] es el título del documento (ejemplo: Estándar de Conteo, Arquitectura).

X.Y.Z indica la versión (X para cambios mayores, Y para medianos y Z para menores).

[extensión] es el tipo de archivo (ejemplo: .docx, .pdf).

Ejemplos: Estandar_de_conteo_v1.0.docx, Manual_de_usuario_2.1.1.pdf.

9 Método para identificar el estado de revisión

9.1 Para el código fuente:

El estado de los archivos de código fuente (.java) se determinará a través del sistema de control de versiones en GitHub. Cada cambio se registra mediante commits.

Las ramas (branches) reflejarán el estado del desarrollo: las ramas /feature/[acción] estarán en estado "en desarrollo", mientras que los pull requests abiertos indicarán "en revisión". Una vez que un pull request es aprobado y fusionado a la rama principal (main), el código pasa a estado "aprobado". Las líneas base (27 de febrero, 27 de marzo, 16 de mayo) marcarán los estados "finalizados" para cada entrega.

9.2 Para los documentos:

Cada documento (ERS, Documento de Arquitectura, Casos de Prueba, Manual de Usuario, Estándar de Codificación y Estándar de Conteo) incluirá una sección de Control de Configuración en la primera página después de la portada, con un Histórico de Versiones que indicará claramente el estado actual (por ejemplo, "borrador", "en revisión", "aprobado", "finalizado") junto con la versión (ej. 1.0.0), fecha y responsable.

Los cambios propuestos se gestionarán en la nube, y el autor del cambio actualizará el Histórico de Cambios para reflejar el estado antes y después de la revisión. Un documento se considerará "aprobado" cuando el equipo lo haya revisado y acordado en su versión definitiva.

10 Líneas base de configuración

- Baseline inicial: Código base para la primera entrega el 27 de febrero.
- Segunda baseline: Versión funcional entregada el 27 de marzo.
- Tercera baseline: Versión final entregada el 16 de mayo.

11 Control de cambios

Para la documentación se usará el siguiente formato para el control de documentación:

11.1 Documentación

11.1.1 Control de configuración

Título:	
Referencia:	
Autor:	
Fecha:	

11.1.2 Histórico de versiones

Versión	Fecha	Estado	Responsable	Nombre del archivo

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

11.1.3 Histórico de cambios

Versión	Fecha	Cambios

11.2 Código fuente:

- El equipo de desarrollo colaborará y gestionará cambios a través de GitHub.

12 Métodos para procesar cambios y concesiones:

12.1 Para los documentos:

- Modificaciones en los documentos deben realizarse en la nube por los integrantes del equipo.
- El autor del cambio debe actualizar el historial de cambios en el documento.

12.2 Para el código fuente:

- Crear una nueva rama para cada funcionalidad.
- Realizar el pull request desde la rama /feature/[acción].
- Asegurar que los cambios pasen las revisiones para poder hacer el merge.