UADY
FACULTAD DE MATEMÁTICAS

Universidad Autónoma de Yucatán

Facultad de Matemáticas

Licenciatura en Ingeniería de Software

Mantenimiento de Software

M.I.T. Carlos Benito Mojica Ruiz

Plan de Gestión de la Configuración

Versión 1.0

Integrantes de Equipo:

Canul Ordoñez, Josué Israel
Garcilazo Cuevas, Mónica
Leo Fernández, José Carlos
Pool Flores, Endrick Alfredo
Rodríguez Coral, Samuel David

CONTROL DE DOCUMENTACIÓN

Título:	Estimación de tamaño para el Proyecto “Contador Azul”
Referencia:	
Autor:	Canul, J. I., Garcilazo, M., Leo, J. C., Pool, E. A., & Rodríguez, S. D.
Fecha:	19 de febrero de 2025

HISTÓRICO DE VERSIONES

Versión	Fecha	Estado	Responsable	Nombre de archivo
1.0	19/02/2025	A	José Carlos Leo Fernández	Plan de Gestión de la Configuración (V1.0).docx

HISTÓRICO DE CAMBIOS

Versión	Fecha	Cambios
1.0	19/02/2025	Realización del Plan de Gestión de la Configuración.

Índice

Introducción..... 3

Identificación de los Ítems de Configuración.....4

1. Criterios para definir qué elementos serán gestionados.....4
2. Estructura y jerarquía de los Ítems de Configuración..... 4
3. Identificación de las herramientas y entornos utilizados.....5

Control de Cambios..... 5

1. Procedimientos para solicitar, evaluar y aprobar cambios.....5
2. Roles y responsabilidades en el control de cambios..... 7

Registro y Seguimiento de la Configuración..... 7

1. Mecanismos de Trazabilidad entre Versiones.....7

Gestión de Liberaciones y Entregas..... 8

1. Procedimientos para la Creación y Distribución de Versiones Finales..... 8
2. Definición de Criterios de Aceptación para Cada Versión.....9
3. Formato y Distribución de Entregables..... 9

Introducción

El presente Plan de Gestión de la Configuración (PGC) establece los lineamientos y procedimientos para la identificación, control de cambios, seguimiento y liberación de versiones del proyecto Contador Azul. Su propósito es proporcionar un marco estructurado que garantice la trazabilidad, integridad y calidad de los elementos de configuración, tanto del código fuente como de la documentación y los ejecutables. Para ello, se adoptan prácticas alineadas con el estándar IEEE 828-2012, que define los procesos esenciales en la gestión de la configuración dentro de la ingeniería de software y sistemas.

Este plan abarca todos los componentes clave del proyecto, incluyendo el código fuente, almacenado en GitHub, donde se gestiona mediante control de versiones y revisión de cambios a través de Pull Requests; la documentación, resguardada en Google Drive, con un control estricto de versiones mediante su historial de cambios; y los ejecutables, generados en formatos .exe para Windows y .jar para macOS, organizados en una estructura de carpetas dentro de cada liberación. Estos elementos serán administrados durante todo el ciclo de vida del proyecto, asegurando que cualquier modificación sea controlada, evaluada y documentada correctamente.

Para fundamentar este plan, se ha tomado como referencia el estándar IEEE 828-2012, que especifica los procesos y actividades fundamentales en la gestión de la configuración de software. Su aplicación en este proyecto permitirá establecer procedimientos claros y eficientes para la identificación y control de versiones, optimizando la calidad del software y garantizando la estabilidad de cada entrega.

Identificación de los Ítems de Configuración

1. Criterios para definir qué elementos serán gestionados

Los elementos que serán gestionados dentro del proyecto Contador Azul se dividen en dos categorías principales:

- Código fuente: Todo el código del proyecto almacenado en GitHub, incluyendo scripts, archivos de configuración y cualquier otro elemento relacionado con el desarrollo del software.
- Documentación: Archivos relacionados con especificaciones, requerimientos, manuales de usuario, reportes y cualquier otro documento de apoyo, almacenados en Google Drive.

Estos ítems de configuración serán gestionados mediante herramientas de control de versiones y procesos de control de cambios definidos en este documento.

2. Estructura y jerarquía de los Ítems de Configuración

La gestión de los ítems de configuración seguirá la siguiente estructura:

- Código Fuente (repositorio en GitHub)
    - Rama principal (main o master): Contendrá la versión estable del software, lista para liberaciones oficiales.
    - Ramas de desarrollo (develop o feature branches): Espacios de trabajo donde se implementan nuevas funcionalidades o correcciones de errores antes de integrarse a la rama principal.
    - Pull Requests (PRs) y revisiones: Todo cambio deberá pasar por revisiones antes de ser fusionado en la rama principal.
    - Etiquetas (tags) y versiones: Se etiquetarán versiones específicas para facilitar la trazabilidad y gestión de versiones.
    - Issues y seguimiento de cambios: Registro de incidencias, mejoras y tareas en el repositorio.
- Documentación (almacenada en Google Drive)
    - Carpeta principal del proyecto: Contendrá toda la documentación relevante del proyecto.

- Control de versiones: Uso de historial de versiones en Google Drive para rastrear cambios en documentos e identificación de cada versión con fechas y descripciones de cambios.

3. Identificación de las herramientas y entornos utilizados

Para la gestión de la configuración, se utilizarán las siguientes herramientas:

GitHub: Para almacenar y gestionar el código fuente, incluyendo control de versiones, revisiones de código y gestión de cambios mediante pull requests.

Google Drive: Para la gestión y almacenamiento de documentos, utilizando su historial de versiones y permisos de acceso para la colaboración.

Control de Cambios

1. Procedimientos para solicitar, evaluar y aprobar cambios

El proceso de control de cambios tiene como objetivo garantizar que cualquier modificación en los ítems de configuración (código y documentación) se realice de manera ordenada, revisada y aprobada antes de su implementación.

Para el código fuente (gestión en GitHub):

1. Solicitud del cambio:
    - Un miembro del equipo crea una nueva rama en GitHub a partir de la rama develop o main.
    - Implementa los cambios en la nueva rama.
    - Abre un Pull Request (PR) hacia la rama de destino, describiendo el cambio realizado.
2. Evaluación del cambio:
    - Al menos un revisor asignado debe revisar el código, asegurándose de que cumpla con los estándares de desarrollo y que no afecte negativamente el proyecto.
    - Se pueden solicitar modificaciones o correcciones antes de la aprobación.

3. Aprobación del cambio:
    - Una vez aprobado por los revisores, el PR se fusiona en la rama correspondiente (develop para cambios en desarrollo o main para cambios listos para producción).
    - Se actualizan los registros de cambios en la documentación del repositorio (changelog o historial de versiones).
4. Seguimiento y documentación:
    - Se etiqueta la nueva versión si corresponde.
    - Se actualizan las tareas o issues relacionadas con el cambio.

Para la documentación (gestión en Google Drive):

1. Solicitud del cambio:
    - Un miembro del equipo propone una modificación en un documento existente o solicita la creación de un nuevo documento.
    - Se agrega un comentario o sugerencia en Google Drive para notificar al equipo del cambio propuesto.
2. Evaluación del cambio:
    - Los responsables del documento revisan la propuesta y deciden si se aprueba o requiere ajustes.
    - Si es necesario, se realizan correcciones antes de la aprobación.
3. Aprobación del cambio:
    - Una vez revisado, el documento es actualizado y la nueva versión se guarda en el historial de versiones de Google Drive.
    - Se notifica al equipo del cambio mediante comentarios o actualización del estado del documento en un espacio compartido.
4. Seguimiento y documentación:
    - Se registra la versión del documento y su estado actual (borrador, aprobado, publicado).
    - Se actualizan enlaces y referencias en otros documentos si es necesario.

2. Roles y responsabilidades en el control de cambios

Rol	Responsabilidades
Desarrollador	- Crear y modificar código en una nueva rama. - Solicitar cambios mediante Pull Requests. - Atender observaciones de los revisores.
Revisor de Código	- Revisar los cambios en los Pull Requests. - Evaluar calidad, impacto y cumplimiento de estándares. - Aprobar o rechazar cambios.
Administrador del Repositorio	- Definir políticas de fusión y gestión de ramas en GitHub. - Mantener la integridad del código en main y develop. - Etiquetar versiones importantes.
Editor de Documentación	- Gestionar cambios en los documentos. - Revisar y aprobar modificaciones en Google Drive. - Mantener el historial de versiones actualizado.
Líder de Proyecto	- Supervisar que los cambios cumplan con los objetivos del proyecto. - Tomar decisiones sobre cambios críticos. - Garantizar la coordinación entre código y documentación.

Registro y Seguimiento de la Configuración

1. Mecanismos de Trazabilidad entre Versiones

Para garantizar la trazabilidad entre versiones, se implementarán los siguientes mecanismos:

Código Fuente:

Control de versiones en GitHub: Uso de commits, pull requests y tags para registrar cambios en el código.

Gestión de issues: Se utilizarán los issues de GitHub para documentar cambios, bugs y mejoras implementadas en cada versión.

Documentación:

Control de versiones en Google Drive: Se habilitará el historial de versiones para rastrear modificaciones en documentos clave.

Identificación de documentos: Se asignará una nomenclatura estándar para cada documento (ej. Estándar de Conteo (V1.0).pdf).

Registro de cambios en documentos: Se incluirá una sección de historial de cambios en cada documento.

Gestión de Liberaciones y Entregas

1. Procedimientos para la Creación y Distribución de Versiones Finales

La generación de versiones ejecutables del software se realizará en dos formatos:

- .jar (Java Archive): Generado con Maven para su ejecución multiplataforma.
- .exe (Windows ejecutable): Creado a partir del .jar mediante Launch4j.

Proceso de Construcción y Distribución:

Construcción del código:

- Se asegura que el código en la rama main esté en su versión más reciente y estable.
- Se ejecuta mvn clean package para generar el .jar.
- Se usa Launch4j para empaquetar el .jar en un .exe para Windows.

Pruebas y validación:

- Se ejecutan pruebas unitarias y de integración.
- Se verifica la funcionalidad del .jar y del .exe en entornos controlados.

Empaquetado y firma:

- Se añaden archivos de soporte (ej. README, manual de usuario).

Publicación y distribución:

- Se suben los archivos finales a un repositorio de distribución (GitHub).

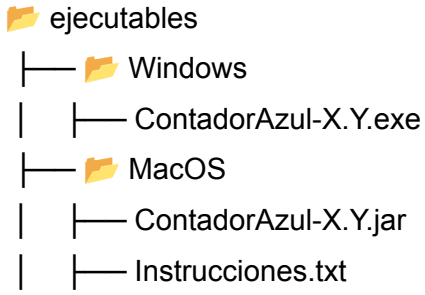
2. Definición de Criterios de Aceptación para Cada Versión

Para que una versión sea liberada, debe cumplir con los siguientes criterios:

- Compilación exitosa: No debe haber errores en la construcción del código.
- Pruebas automatizadas aprobadas: Todas las pruebas unitarias e integración deben ejecutarse correctamente.
- Verificación manual: Pruebas funcionales realizadas por el equipo de desarrollo y/o QA.
- Documentación actualizada: Se debe incluir un registro de cambios en el CHANGELOG.md y actualizar los manuales de usuario si es necesario.
- Validación por parte del equipo: Aprobación final por el equipo antes de la publicación.

3. Formato y Distribución de Entregables

Cada versión liberada se organizará en una carpeta de ejecutables, la cual contendrá subcarpetas para cada sistema operativo:



Contenido de las carpetas:

Windows: Contendrá el archivo ejecutable .exe, listo para instalación y uso en sistemas Windows.

MacOS: Incluirá el archivo .jar, ejecutable en sistemas MacOS, junto con un archivo de texto Instrucciones.txt que proporcionará detalles sobre cómo ejecutar el .jar en MacOS.

Referencias

Institute of Electrical and Electronics Engineers. (2012). IEEE Standard for Configuration Management in Systems and Software Engineering (IEEE Std 828-2012). IEEE. https://doi.org/10.1109/IEEESTD.2012.6170935