AZERTY

Trucardo

Configuration Management Plan

Autores:

- ➢ Colque Ventura, Santiago Agustín
- ➢ Fernandez, Ignacio Javier
- ➢ Lopez Paviolo, Franco Marcelo

Fecha:

- ➢ 30 de abril de 2020

Versión 1.0.1
Grupo AZERTY
Año 2020

ÍNDICE

HISTORIAL DE CAMBIOS

Versión	Fecha	Cambios
1.0.0	**30 abril 2020**	**Primera versión**
1.0.1	**7 mayo 2020**	**Se agregó índice**
1.0.2	**24 de mayo 2020**	**Se agregaron cambios a las secciones 2.2, 4.2 y 5.2**

ACRÓNIMOS/GLOSARIO

Acrónimo	Descripción
CCB	Change Control Board
CM	Configuration Management
SCM	Software Configuration Management
GPCM	Global Project Configuration Manager
CR	Change Request

1 INTRODUCCIÓN

1.1 PROPÓSITO Y ALCANCE DEL PLAN

Este documento plantea el plan de Gestión de las Configuraciones (Configuration Management Plan) del proyecto Trucardo. El objetivo del plan de CM es el controlar la configuración de los requerimientos, documentos, software y herramientas utilizadas en este proyecto. SCM es el proceso mediante el cual se identifican las herramientas y métodos de control a lo largo del desarrollo y uso.

1.2 PROPÓSITO DEL PLAN DE GESTIÓN DE CONFIGURACIONES

- Mantener la integridad del proyecto
- Definir los roles del equipo dentro del proyecto
- Crear un historial del desarrollo del proyecto
- Mantener informados a los integrantes del proyecto sobre el estado actual

1.3 HERRAMIENTAS PARA LA ADMINISTRACIÓN DE CONFIGURACIONES

Herramienta	Descripción	Forma de acceso
Trucardo_GitHub	Control de versiones del proyecto	https://github.com/francomlopez/Trucardo
Trucardo_Circleci	Control de Integración Continua para realizar tests	https://app.circleci.com/pipelines/github/francomlopez/Trucardo
Trucardo_Monday	Gestión de tareas del proyecto	https://azerty48.monday.com/boards/550122241
Trucardo_GitHub-issues	Gestión de defectos a corregir	https://github.com/ignacioFernandez1/Trucardo/issues

1.4 ROLES Y RESPONSABILIDADES

Rol	Descripción	Responsable
Gestor global de configuración (GPCM)	• Responsable de asegurar que se cumpla con lo establecido en el plan de gestion de configuracion. • Promover el uso efectivo de las herramientas elegidas para la realización del proyecto • Monitorear y reportar los cambios en las branches del proyecto • Monitorear y controlar el correcto uso de labels en las distintas versiones • Aprobar cambios estructurales en el plan de gestión de configuración.	Ignacio Fernandez
Coordinador de configuración	• Asegurar que todos los elementos de configuración están registrados de forma adecuada en el plan de configuración. • Asegurar la consistencia e integridad de los datos del plan de configuración y la estructura del sistema a través de la ejecución de procedimientos de verificación y auditoría. • Reportar cualquier discrepancia o no conformidad en los elementos de configuración al gestor de configuración. • Participar en la mejora continua del proceso de gestión de configuración.	Franco Lopez

Responsable de elementos de configuración	• Asegurar que los elementos de configuración de los que es responsable están registrados en el plan de configuración con el estado y datos de configuración apropiados. • Verificar que los cambios sobre los elementos de configuración siguen el proceso de cambios definido. • Trabajar conjuntamente con el gestor de configuración para identificar las causas de cualquier discrepancia identificada en las auditorías e implementar las acciones correctivas.	Santiago Colque

2 ESQUEMA DE DIRECTORIOS

2.1 ESTRUCTURA DE DIRECTORIOS Y PROPÓSITO DE CADA UNO

Nuestro proyecto se almacenará dentro de un directorio base llamada Trucardo.Dentro del mismo podremos encontrar:

- Trucardo\Documentacion : Planes generales y archivos relevantes que aplican para todos los releases.
- Trucardo\Releases: dentro de este directorio se encontrarán todos los releases realizados con su documentación correspondiente.
- Trucardo\src: se encuentra el código en Java de nuestro proyecto y los test correspondientes.

2.2 NORMAS DE ETIQUETADO Y NOMBRAMIENTO DE ARCHIVOS

Para las diferentes versiones de los documentos seguiremos un estándar del tipo Semantic Versioning. El cual se basa en lo siguiente:

- Las versiones siguen el tipo MAJOR.MINOR.PATCH las cuales aumentan según lo siguiente:

- PATCH: Si se realiza un bug fix.
- MINOR: Se introduce nueva funcionalidad y mantiene la compatibilidad con versiones previas.
- MAJOR: Se introducen nuevas funcionalidades que no son compatibles con versiones previas o si se cambia por completo al proyecto.

- Cuando se incrementa algún número del número de versión, los números a su derecha se reinician. Por ejemplo: si se tiene la version 3.6.5 y se realiza un change MINOR, próxima versión será 3.7.0.
- Adicionalmente existen 3 extensiones al número de versión con la siguiente forma: 3.18.9-*extension.ID.* Estas son:
  - alpha: 3.18.9-alpha.1
  - beta: 3.18.9-beta. Esta versión es la que sigue a la alpha
  - rc: 3.18.9-rc. RC quiere decir Release Candidate. Esta versión es la anterior a una release y la que le sigue a la beta.

3 GESTIÓN DE LA CONFIGURACIÓN DEL CÓDIGO

3.1 ESQUEMA DE RAMAS

Se utilizará un esquema con cuatro tipos de rama. Estos son: *master*, *stable*, *feature, bug* y *hotfixes*.

Las ramas *master* y *stable* serán las principales, y perpetuarán indefinidamente.

La rama *stable* se utilizará para versiones estables y releases, mientras que la rama *master* será la rama principal. Los desarrolladores realizarán merges y branches tomando como punto de partida a esta rama.

Las ramas *feature* tendrán lugar a la hora de desarrollar una nueva función o realizar una mejora que potencialmente acabarán incorporándose al proyecto. Una vez integrada la nueva funcionalidad a la rama master, esta terminará su vida útil.

Las ramas *bug* se crearán al descubrir un bug en la rama *master*, y al igual que en *feature*, llegarán a su fin una vez realizado el merge correspondiente hacia *master.*

Una rama *hotfix* surge de la necesidad de actuar inmediatamente al tomar conocimiento de un comportamiento indeseado en un release, parte desde la rama *stable* de

manera tal que posibilita trabajar en el fix y en *master* al mismo tiempo.

3.2 POLÍTICA DE ETIQUETADO DE RAMAS

Si la rama no existe todavía, se creará una nueva rama con un número de id de la forma <tipo_de_branch-id>. Por ejemplo *feature-1* o *bug-2*.

3.3 POLÍTICA DE FUSIÓN DE ARCHIVOS

La política de fusión de cada rama se detalla a continuación:

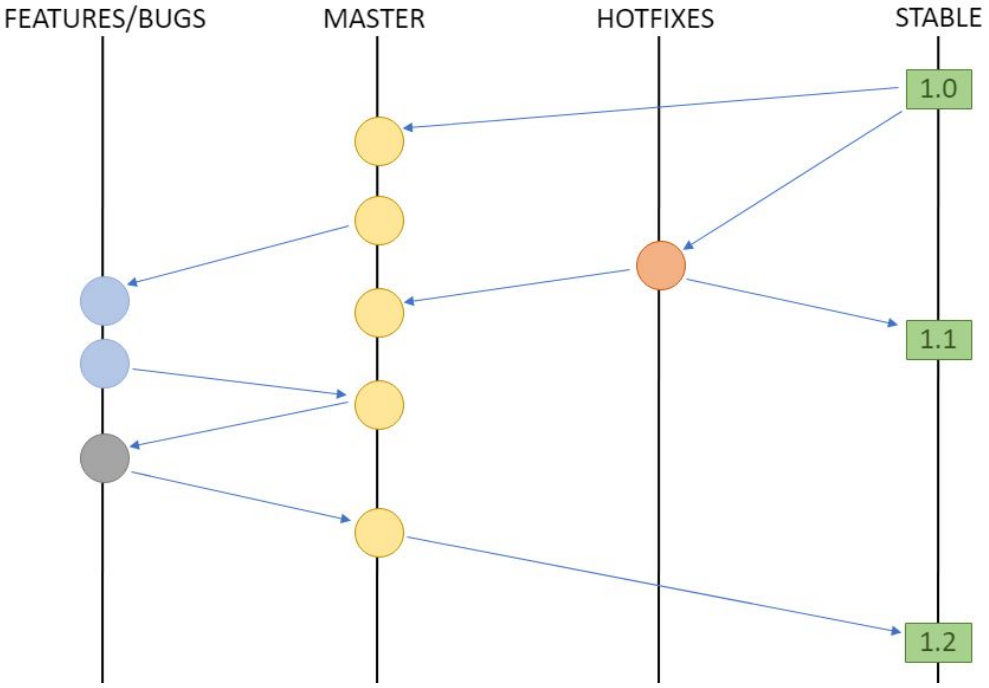
- *feature:*
  - Debe salir de la rama *master*
  - Debe mergear a la rama *master*
  - Cualquier cambio en *master* debe ser reflejado en *feature* antes de volver a mergear a *master*

- *bug:*
  - Debe salir de la rama *master*
  - Debe mergear a la rama *master*
  - Cualquier cambio en *master* debe ser reflejado en *bug* antes de volver a mergear a *master*
- *hotfix:*
  - Debe salir de la rama *stable*
  - Debe mergear a la rama *master* y a la rama *stable*

Además, cuando el código de la rama *master* es estable y apto para release, este debe ser mergeado hacia *stable.*

No se podrá mergear a *master* si no se pasan los test que son corridos en Circleci.

Todo esto se puede ver de una manera más gráfica, en el ejemplo expuesto en el diagrama siguiente:


FEATURES/BUGS
MASTER
HOTFIXES
STABLE
1.0
1.1
1.2

4 GESTIÓN DE CAMBIOS

4.1 CHANGE CONTROL BOARD (CCB)

4.1.1 INTRODUCCIÓN Y OBJETIVOS

En este comité se revisarán los cambios a introducir en el proyecto ya sea en la documentación o en el código fuente del producto. El objetivo de este comité es aplicar los cambios necesarios en el proyecto con la aprobación de todos sus miembros.

4.1.2 MIEMBROS

Rol	Responsabilidad	Responsable
CCB Líder	Responsable de manejar la herramienta de gestión de defectos (GitHub Issues), documentando específicamente sobre cada problema que se trabaja. Dar la aprobación final al cambio a realizar en el proyecto y documentando la finalización del problema.	Franco Lopez Paviolo
Gestor de problemas	Encargado de traer los problemas a las reuniones del comité con todas las especificaciones necesarias.	Santiago Colque
Gestor global de configuración	Asegurar que el procedimiento de corrección de errores cumpla con lo dispuesto en el plan de gestión de configuración.	Ignacio Fernandez

4.1.3 FRECUENCIA DE REUNIÓN DE TRABAJO

Las reuniones se realizarán 2 o más veces a la semana y a medida que el proyecto avance, la frecuencia podría llegar a aumentar.

4.2 PROCESO DE CONTROL DE CAMBIOS

El proceso de control de cambios deberá seguir un proceso estructurado por el cual se somete al CR a un análisis. Este análisis consta de una serie de pasos a seguir:

1. Análisis del problema: En esta primer etapa se identifica el cambio solicitado, se analiza su veracidad y a su vez la solución que se plantea. Si el problema se considera válido, se procede al siguiente paso.
2. Análisis de implementación: En esta etapa se evalúa la implementación del cambio y sus costos. Se debe verificar el impacto del cambio en el resto del sistema observando cómo afecta a los componentes del mismo, analizar si hacer el cambio genera nuevos cambios a realizar en otras partes del sistema, lo cual aumentará el costo de la implementación. Teniendo en cuenta este costo de cambio se analizan los beneficios y consecuencias que el cambio traería, para decidir si es recomendable aprobar el cambio.
3. Implementación del cambio: Una vez aprobado el cambio, se prosigue a implementar dicho cambio al sistema. En la mayoría de las situaciones se deberá modificar también otras componentes del sistema para adaptarlos según el cambio realizado. Finalmente, se testea el software para garantizar que no se hayan introducido nuevos bugs y que nuestro sistema funcione de acuerdo al cambio realizado. En caso de que fallen los tests, se vuelve a modificar el software hasta que estos pasen exitosamente. Una vez que esto último suceda, se pasa la siguiente etapa.
4. Aceptación del cambio: Como último paso, se registra el cambio realizado, se actualiza la versión del sistema siguiendo las reglas de etiquetado y se cierra la CR.

4.3 HERRAMIENTA DE GESTIÓN DE CAMBIOS

Como herramienta de gestión de cambios se utilizara GitHub-Issues.

5 GESTIÓN DE ENTREGAS

5.1 FORMATO DE ENTREGA DE RELEASES

Para el formato de entrega de releases se incluirá en la entrega lo siguiente:

- .jar del programa
- Archivo README.txt el cual incluye información sobre versión de release, ayuda con la instalación del software y bugs conocidos.

5.2 FORMATO DE ENTREGA DEL INSTALADOR

El formato de entrega del instalador será a través de un archivo comprimido .zip el cual incluirá el .jar del programa, además se incluirá un README.txt con las instrucciones para la ejecución del programa, como así también un documento con las reglas del juego para que los jugadores principiantes puedan entender cómo se juega.

5.3 INSTRUCCIONES MÍNIMAS DE INSTALADOR

Aún no se ha determinado cómo será el proceso de instalación. Las instrucciones serán incluidas en un documento de texto adjunto a la primera entrega.