2021

Testigos de Turing

PLAN DE GESTIÓN DE CONFIGURACIONES

Autores:

- Ciordia Cantarella , Francisco
- Martinez, Brenda Sofia
- Fernandez, Santiago
- Rao, Maximiliano
- Villane, Santiago

Tabla de contenido

HISTORIAL DE REVISIONES

Versión	Fecha	Resumen de cambios	Autores
0.1.0	28/04/2021	Documento inicial	Ciordia Cantarella Francisco Martinez Sofia Rao Maximiliano Fernandez Santiago Villane Santiago
1.0.0	29/04/2021	Documento mejorado	Ciordia Cantarella Francisco
1.1.0	29/04/2021	Incorporación Índice	Rao Maximiliano
1.1.1	29/04/2021	Informe final	Ciordia Cantarella Francisco Martinez Sofia Rao Maximiliano Fernandez Santiago Villane Santiag
1.1.2	22/05/2021	Corrección de errores #1	Francisco Ciordia Fernandez Santiago

REGISTRO DE APROBACIÓN

Gerente de configuración (CM)	Fecha	Firma

Subgerentes de manejo de la configuración	Fecha	Firma

1. INTRODUCCIÓN

1.1. .PROPÓSITO Y ALCANCE DEL PLAN

El presente documento expone el Plan de Gestión de Configuraciones realizado para el proyecto “LUDO MATIC”. Mediante este plan se busca poner en conocimiento las políticas, estrategias, herramientas y métodos empleados para el manejo del software producido.

1.2. PROPÓSITO DEL PLAN DE GESTIÓN DE CONFIGURACIONES

· Definir políticas y procesos de la administración de configuración; y explicitar los fundamentos de las mismas.

. Mantener la integridad el proyecto.

· Establecer cuáles son los roles en el CCB.

· Definir herramientas de gestión de versiones y entrega.

· Exponer los principios para la construcción del sistema que garanticen la calidad requerida y pretendida.

.Crear un historial del desarrollo del proyecto y mantener informados a los integrantes sobre el estado actual del mismo.

1.3. HERRAMIENTAS PARA LA ADMINISTRACIÓN DE CONFIGURACIONES

Herramienta/Proceso	Propósito
GitLab	Control de versiones. Integración continua
GitHub_Issues	Herramienta de gestión de defectos a corregir
Monday	Herramienta de gestión de tareas

2. NORMAS DE ETIQUETADO DE ARCHIVO Y DIRECTORIO

2.1. TAG DE DIRECTORIOS

Estructura de directorios y fines.

Para la documentación, código fuente y los releases se realizaron las siguientes denominaciones:

- LudoM.Documento: Directorio de planes generales, documentos de requerimientos, test y diseño.
- LudoM.Releases: Directorio donde se encuentran los releases.
- LudoM.src: Directorio donde se ubica el código fuente del proyecto y sus diversos testings.

2.2. TAG DE VERSIONADO DE ARCHIVOS

Para el versionado utilizamos el tipo estándar, separando por puntos, según se trate de un cambio menor, mayor o un simple fix, popularmente conocido como estándar Semantic versioning.

<Nombre_de_archivo>.MAYOR.MENOR.FIX-*extension.ID*.exe

Donde:

- FIX: Al parchear un bug.
- MENOR: Al agregar una funcionalidad sin tener que cambiar el resto del código.
- MAYOR: Al agregar funcionalidad la cual no era compatible previamente y se hicieron cambios en el código base respecto a la versión anterior.

Los cambios son crecientes y tras cierto número de cambios de menor jerarquía se considera uno de mayor, es decir, cada 10 de menor se le suma 1 a mayor seteado a menor en 0 nuevamente, siendo el nivel de jerarquía de derecha a izquierda siendo FIX la de menor. Además, un cambio en la jerarquía mayor, también setea a las anteriores en 0, es decir; pasamos de Main_2.3.4 -> a un Main_3.0.0 de un solo salto.

extension.ID: Etiqueta que se agrega al nivel de testeo que tenga el proyecto:

- Alpha:Tipo de usuario dentro del grupo de testeo.
- Beta: Tipo de usuario dentro del grupo de trabajo ajena a la formulación del código
- RC (Release Candidate): Tipo de usuario ajeno al grupo que va a hacer distintas pruebas de usuario.

Cabe destacar que los términos Alpha, Beta y RC hace referencia al ciclo de vida de lanzamiento de un software y que son etapas del desarrollo como tal, que se corresponden a un estado del producto en si.

3. CHANGE CONTROL BOARD

3.1. INTRODUCCIÓN Y OBJETIVOS

El Comité de Control de Cambios es un grupo dentro del equipo que se encarga de evaluar, aprobar o rechazar los pedidos de cambios realizados por diversos agentes.

Tiene como fin recibir los pedidos de cambios y evaluar su autorización para luego planear y generar la respectiva documentación y código.

La toma de decisiones tiene como criterio el cumplimiento de los requerimientos de calidad y el correcto desenvolvimiento de cada prueba.

3.2. EQUIPO DE TRABAJO

Posición dentro del CCB	Titular	Suplente	Función
Director del CCB	Fernández, Santiago	Ciordia, Francisco	• Aprobación o modificación de cambios.

			• Organizador de reuniones. • Manejo del CCB. • Moderador.
Gerente de manejo de configuraciones (GMC)	Rao, Maximiliano	Ciordia, Francisco	• Responsable en general de tareas de planeamiento, diseño de las políticas de control de seguimiento y su actualización. • Veedor de cumplimiento de lo anterior
Gerente de ingeniería o de release (GI)	Villane, Santiago	Martínez, Brenda Sofía	• Evaluación del costo en cuanto a cambios de infraestructura del sistema, fechas de entrega, etc.
Gerente de coordinación (GC)	Martínez, Brenda Sofía	.	• Se encarga de las peticiones de cambio mediante change request forms y logs. • Documenta y traduce a modo formal los cambios.
Gerente de pruebas	Ciordia, Francisco		• Aplica el plan de requerimientos para contrastar con lo obtenido. • Organizar las pruebas

			• Definir las actividades a implementar como estrategia de prueba (test) • Evalúa impactos del cambio en la calidad de la entrega y el calendario de testing

3.3. REUNIONES DE TRABAJO

Las reuniones se harán de manera remota los días jueves a las 17:00 hs y fines de semana si es necesario discutir alguna cuestión extra. Además de los medios de comunicación personales habituales

3.4. PROCESO DE CONTROL DE CAMBIOS

Este proceso consta de las siguientes etapas:

1) Inicio: Un consumidor o miembro del equipo genera una solicitud de cambio la cual se registra para ser tenida en cuenta más tarde

2) Análisis: El soporte de cambios (GC) analiza la solicitud, aca la califican de valida o invalida, de ser inválida o redundante esta se desecha y se cierra la solicitud. Si se acepta pasa al equipo de desarrollo.

3) Desarrollo: En el equipo de desarrollo(GI) se analiza la implementación, a partir de la misma se hace una análisis de costos beneficios, también se tiene en cuenta a la cantidad de usuarios que afectó y cómo impacta esto en el momento de desarrollo actual, a la vez de las consecuencias de no hacerse el cambio.

4) CCB: Acá es cuando el Change Control Board debe hacer un análisis final de la solicitud de cambio con todas las herramientas dadas por el equipo de desarrollo. En esta instancia la solicitud puede ser aceptada, postergada o cerrada, ya que hay que tener en cuenta que las solicitudes de cambios van a darse en numerosas cantidades y es tarea del CCB elegir cuales conviene llevar adelante ahora, más adelante o no llevarlas a cabo por los efectos del mismo.

5) Modificación del software: Cuando el CCB decidió finalmente llevar a cabo la solicitud de cambio, esta vuelve al equipo de desarrollo la cual va a generar los cambios del software, y los va a testear verificando la integridad del proyecto. Una vez pasen las pruebas, se cierra la solicitud de cambio.

4. SOURCE CODE CONFIGURATION MANAGEMENT

4.1. ESQUEMA DE RAMAS Y POLÍTICAS DE FUSIÓN DE ARCHIVOS

Se procederá a especificar las ramas del desarrollo del proyecto. Se encontraran 4 ramas:

First, Improvement, Bugs, Solid y Rapid Repair.

- ➔ Las rama **First** es la rama principal del proyecto. Estará indefinidamente en el proyecto y es el punto de partida de todas las ramas. De esta rama, sale la versión final del proyecto.

- ➔ La rama **Improvement** es la rama destinada a nuevas funciones o nuevas implementaciones para el proyecto. Cuando dichas funciones son implementadas, esta rama termina su vida útil.

➔ La rama **Bugs** es la rama que tiene su aparición cuando se detectan errores en la rama First o Improvement. Como se mencionó anteriormente, parte de la rama First, luego mergea a First y termina su vida útil una vez que los bugs han sido corregidos o eliminados

➔ La rama **Solid** se utilizará para versiones estables y para releases. Esta rama será el punto de partida para realizar merges y branches.

➔ La rama **Fast Repair** se utilizará para corregir errores encontrados en el código, los cuales no pueden esperar una nueva versión para ser corregidos.

POLÍTICAS DE FUSIÓN DE ARCHIVOS

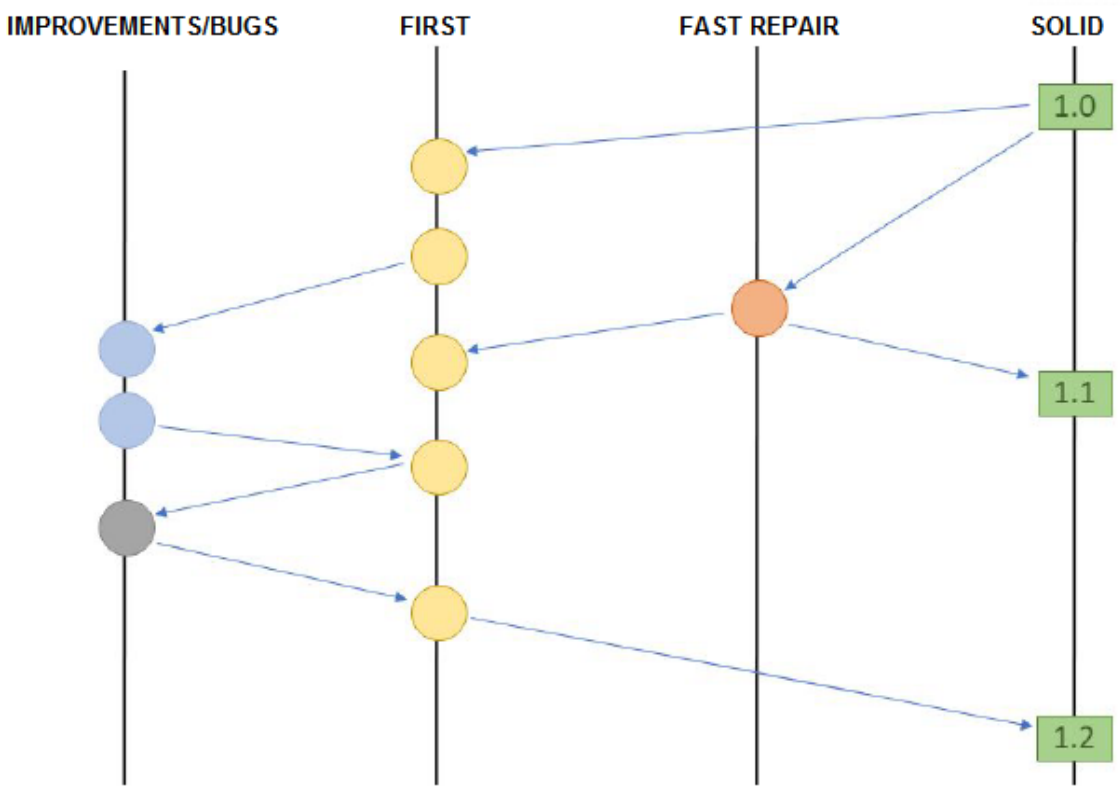
Ramas	*Salida*	*Merge*
Improvement	Saldrá de First	Mergeara a First
Bugs	Saldrá de First	Mergeara a First
Fast Repair	Saldrá de Solid	Mergeara a First y a Solid

➢ Cuando el código de la rama First es estable y listo para release, mergeara a la rama Solid.

ETIQUETADO DE RAMAS:

La política de etiquetado será de la forma <tipo_de_rama- id> donde id es un número de identificación de dicha rama. Por ejemplo: Bugs-1.

DIAGRAMA DE RAMAS



5. MANTENIMIENTO DE CM PLAN

El responsable de monitorear el Plan de Configuración es el GMC. Se hará una revisión del Plan de Configuración al comienzo de cada iteración. En caso de que haya modificaciones se comunicarán por correo a todos los integrantes.

6. BACKUP Y RECOVERY EN CASO DE DESASTRE

Se tendrá una política de Backup semanal, los martes de cada semana, el GMC será el encargado de realizar la un “clone” de todo el repositorio de GitHub utilizado de esta forma tendremos una copia local del mismo. Este Backup tendrá una política de retención de 2 semanas.

Para la restauración el GMC tendrá un tiempo máximo de 1 hora para recuperar el repositorio completo.

7. RELEASE MANAGEMENT

Los “releases” o lanzamientos estarán presente en la rama principal (master branch).

Para cada release se entregar un archivo .zip donde se incluirá lo siguiente:

- El ejecutable del programa .cmd

- El programa en un archivo .jar

- Archivo .pdf de instrucciones de uso

- Archivo README.txt el cual incluye información sobre versión de release,

ayuda con la instalación del software y bugs conocidos.