SuiGeneris - Interno

WPMTest

PLAN DE GESTIÓN DE CONFIGURACIONES

Autores:

* Cancinos, José
* Giraudo, Juan Pablo
* Oliva, Nahuel

Versión del documento: ¿?????????????

# HISTORIAL DE REVISIONES

|  |  |  |  |
| --- | --- | --- | --- |
| Versión | Fecha | Resumen de cambios | Autores |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# REGISTRO DE APROBACIÓN

Lista de miembros del equipo cuya aprobación debe ser explicitada para todo cambio significativo en este plan:

Aclaración:

* Aprobaciones no son necesarias en caso de no ser significativas.
* Aprobación sobre el plan son solo necesarias si el cambio es realizado por otra persona distinta al Administrador de Configuraciones.

|  |  |  |
| --- | --- | --- |
| Gerente de ingeniería (engineering manager) | Fecha | Firma |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Facilitador de proyecto (Scrum Master) | Fecha | Firma |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Gerente de entregas  (release manager) | Fecha | Firma |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Gerente de proceso  (process manager) | Fecha | Firma |
|  |  |  |

|  |  |  |
| --- | --- | --- |
| Gerentes de configuración (CM) | Fecha | Firma |
|  |  |  |

# ÍNDICE

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | INTRODUCCIÓN | | ………………. |  |
|  | 1.1 |  | ………………. |  |
|  |  |  | ………………. |  |
|  |  |  | ………………. |  |
|  |  |  | ………………. |  |
|  |  |  | ………………. |  |
|  |  |  | ………………. |  |
|  |  |  | ………………. |  |

# INTRODUCCIÓN

* 1. PROPÓSITO Y ALCANCE DEL PLAN

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

* 1. PROPÓSITO DEL PLAN DE GESTIÓN DE CONFIGURACIONES
* Definir políticas y procesos de la administración de configuración; y explicitar los fundamentos de las mismas.
* 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.
  1. GLOSARIO

(revistar si usamos o no)

* 1. HERRAMIENTAS PARA LA ADMINISTRACIÓN DE CONFIGURACIONES

|  |  |
| --- | --- |
| Herramienta/Proceso | Propósito |
| Android Studio | Herramienta de desarrollo de aplicaciones para Android. IDE. |
| GitLab | Control de versiones.  Integración continua |
| VisualParadigm | Herramienta gráfica para diagramas UML |
| GitLab CI/CD | Herramienta de integración continua y monitoreo de cada rama |
| JUnit | Herramienta de pruebas automatizadas |
| Maven | Herramienta de construcción automática |
| Microsoft Word | Editor de texto para la realización de los informes |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

**1.4.1)** DIRECCIÓN Y FORMA DE ACCESO A LA HERRAMIENTA DE CONTROL DE VERSIONES

Se emplea la herramienta GitLab.

La dirección del mismo es: <https://gitlab.com/Nadaol/typing_app>

Se accede mediante petición de acceso al propietario del repositorio del proyecto ya que el mismo se ha declarado como “privado”. Es requerido poseer una cuenta en GitLab.

1.4.2) DIRECCIÓN Y FORMA DE ACCESO A LA HERRAMIENTA DE INTEGRACIÓN CONTINUA

Se emplea la herramienta integrada a GitLab de integración continua y entrega continua – GitLab CI/CD.

La dirección del mismo es el propio directorio que utilizaremos. Para mayor información, el enlace de GitLab CI/CD es: <https://docs.gitlab.com/ee/ci/README.html>

Se accede con una cuenta GitLab. Si bien es una funcionalidad de pago, ésta es brindada de forma gratuita para proyectos open source y estudiantiles.

**1.4.3)** DIRECCIÓN Y FORMA DE ACCESO A LA HERRAMIENTA DE GESTIÓN DE DEFECTOS

JUnit.

Se integrará junto a GitLab para el testing del código y la realización de pruebas automáticas.

Acceso mediante GitLab.

Mayor información en documentación de pruebas JUnit: https://docs.gitlab.com/ee/ci/junit\_test\_reports.html

**1.4.4)** DIRECCIÓN Y FORMA DE ACCESO A LA HERRAMIENTA DE SCRIPT DE CONSTRUCCIÓN AUTOMATIZADA

Maven.

Permite colocar un modelo del proyecto de software que se construirá. Incluir todos los módulos y componentes externos. Definir el orden de construcción de los elementos del código.

https://docs.gitlab.com/ee/user/project/packages/maven\_repository.html

<https://maven.apache.org/>

# 4) Esquema de directorios y propósito de cada uno.

ESQUEMA DE DIRECTORIOS

|  |  |
| --- | --- |
| README | ./readme |
| Contiene el archivo “README.md”. Documento que contiene la información esencial del proyecto (nombre, descripción, autores, licencias, etc) | |

|  |  |
| --- | --- |
| LICENCIAS | ./licencias |
| Contiene las declaraciones completas sobre licencias y derechos de copyright sobre el proyecto. | |

|  |  |
| --- | --- |
| VERSION | ./version |
| Contiene cada paquete final de las versiones obtenidas del software | |

|  |  |
| --- | --- |
| LANZAMIENTOS (RELEASE) | ./release |
| Contiene paquetes, notas de lanzamiento y archivos binarios | |

|  |  |
| --- | --- |
| REQUERIMIENTOS | ./requerimientos |
| Archivos con declaraciones de requerimientos especificados para el proyecto. | |

|  |  |
| --- | --- |
| DOCUMENTACIÓN | ./docs |
| Documentación propia del proyecto. Informes internos. Recursos externos. | |

|  |  |
| --- | --- |
| PRUEBAS | ./pruebas |
| Suite o conjunto de pruebas unitarias | |

|  |  |
| --- | --- |
| CÓDIGO FUENTE | ./src |
| Contiene el código fuente del proyecto. Se divide en subdirectorios según paquetes o “namespaces”. | |

|  |  |
| --- | --- |
| ARCHIVOS | ./archivos |
| Archivos varios del proyecto, como archivos de configuración de herramientas, comandos, archivos de texto, etc. | |

|  |  |
| --- | --- |
| DISEÑO | ./diseno |
| Archivos varios | |

|  |  |
| --- | --- |
| HERRAMIENTAS | ./herramientas |
| Archivos varios | |

|  |  |
| --- | --- |
| BASE DE DATOS | ./basedatos |
| Archivos varios | |

# 5) Normas de etiquetado y de nombramiento de los archivos.

POLÍTICAS:

* Uso de guiones (-) para separar etiquetas del nombre.
* Uso de puntos (.) para separar palabras o números.
* Nunca usar espacios o guiones bajos.
* No usar caracteres especiales.
* Usar letras minúsculas.
* Uso de “versionado de software” (SemVer - semantic versioning, en inglés) para indicar el nivel de versión del archivo[[1]](#footnote-1)

SECCIONES DEL NOMBRE:

1. Acrónimo del proyecto: WPM
2. Nombre del archivo
3. Fecha: en formato
4. Numero de versión: X.Y.Z

* Posición 1: indica cambios de gran importancia. Cambios en la API.
* Posición 2: indica cambios menores. Nuevas funcionalidades compatibles con la presente versión mayor.
* Posición 3: parches al software. Solución a bugs.

1. Estado del archivo:

* Final: final
* Estable: stable
* Inestable: un

1. Extensión del archivo

EJEMPLO:

wpm-plan.gestion.configuraciones-2019.04.18-1.0.0-final.pdf

# 6) Plan del esquema de ramas a usar.

# 7) Políticas de fusión de archivos y de etiquetado de acuerdo al progreso de calidad en los entregables.

# 8) Forma de entrega de los “releases”, instrucciones mínimas de instalación y formato de entrega.

# 9) Change Control Board. Se debe incluir el propósito, la lista y forma de los integrantes del equipo y su rol en la CCB, la periodicidad de las reuniones, etcétera.

# 10) Herramienta de seguimiento de defectos usada para reportar los defectos descubiertos y su estado. Forma de acceso y dirección.

# 11) Cualquier otra información relevante.

# 2) …………….

………………..

……………….

# 4) COMITÉ DE CONTROL DE CAMBIOS - CCB (*CHANGE CONTROL BOARD*)

El Change Control Board (Comité de Control de Cambios, en español) es el grupo encargado de evaluar, aprobar o rechazar los pedidos de cambios realizados por diversos agentes internos o externos (clientes).

El comité busca que todo cambio sea considerado por todas las partes y lograr su total autorización antes de su implementación en los planes, documentos y el código de la aplicación.

Se monitorea y controla las peticiones de cambio para establecer luego las bases de los requisitos de configuración.

Las decisiones deben realizarse en base de actividades que permitan asegurar la calidad de producto y el buen avance del objetivo en cada ciclo de prueba.

El CCB debe cumplir con reuniones de trabajo de forma periódica en dónde se definen las políticas de trabajo.

**MIEMBROS[[2]](#footnote-2):**

|  |  |  |  |
| --- | --- | --- | --- |
| Posición dentro del CCB | Titular | Suplente | Función |
| Director del CCB |  |  | * Aprobación o modificación de cambios * Aprobación de gastos * Organización de reuniones * Manejo del CCB * Difusor de las decisiones del CCB |
| Gerente de manejo de configuraciones  (GMC) |  |  | * Responsable en general de tareas de planeamiento, diseño, gastos, contratación y dirección |
| Gerente de ingeniería o de release  (GI) |  |  | * Implementar y administrar las decisiones técnicas del GMC * Evaluación en cambios de infraestructura, fechas de entrega, etc. |
| Gerente de coordinación  (GI) |  |  | * Implementar y administrar las decisiones procedimientales del GMC * Change request forms * Change control logs * Prepara y organiza el temario de las reuniones |
| Gerente de pruebas |  |  | * Aplica el plan de requerimientos para contrastar con lo obtenido. * Organizar los grupos de test * 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 |
| Gerente de producto |  |  | * Enlace con el canal de clientes y ventas. * Responsable de la sustentabilidad financiera del proyecto. * Evaluación del impacto comercial y técnico del cambio propuesto |

FRECUENCIA DE REUNIÓN DE TRABAJO

|  |  |
| --- | --- |
| MOTIVO | FRECUENCIA |
| Reunión periódica del comité | Martes y Jueves. |

1. Versionado de Software – Wikipedia. https://es.wikipedia.org/wiki/Versionado\_de\_software [↑](#footnote-ref-1)
2. “Software Configuration Management Implementation Roadmap” – Mario E. Moreira. 2004 [↑](#footnote-ref-2)