Grupo Go

Trabajo Final
Plan de Gestión de Configuraciones
(CM_Plan)

Autores(s):	Fabián Wolfmann Nicolás Passaglia Agustín Colazo
Versión del documento:	1.2.0

Tabla de contenido

SECCION 1

Introducción

Propósito y panorama

Este documento contiene el Plan de Gestión de Configuraciones para el proyecto Final. El objetivo de este informe de configuraciones, es poner en manifiesto la manera a trabajar en este proyecto respecto a los requisitos, documentos, software y todas las herramientas utilizadas.

En este proyecto se usaran ciertas herramientas y en este informe se detallara la manera de utilizar estas herramientas durante este proyecto.

El objetivo principal del plan de configuraciones es poder asegurarse una forma de ágil y correcto de desarrollador código.

Objetivo de las practicas de Gestión de Configuraciones

- Asegurar un criterio común.
- Mantener comunicación entre los integrantes del proyecto.
- Llevar control de todos los cambios hechos y propuestos.
- Garantizar el desarrollo de software consistente.

Herramientas de CM

HERRAMIENTA	PROPOCITO	LINK DE LA HERRAMIENTA
GITHUB	Utilizado para el control de versiones, servidor en la nube.	https://github.com/fawolfmann/IngSoftwareFinalGo
TRAVISCI	Herramienta para la continua integración del proyecto.	https://travis-ci.org/fawolfmann/IngSoftwareFinalGo
GRADLE	Herramienta utilizada en la integración continua, donde se codifica la compilación y corre los test necesario y configurados.	Archivo dentro del repositorio
GITHUB	Herramienta de gestión de defectos	https://github.com/fawolfmann/IngSoftwareFinalGo/issues

SECCION 2

Roles en la Gestión de Configuración

Administrador en la configuración del proyecto

Este rol es asignado a una sola persona que se encargara de hacer el seguimiento a la rama central y de la aprobación y lanzamiento de ramas criticas (ver sección 6), la persona encargada de esto es el Administrador de Configuración (AC)

Responsabilidades en la Gestión de Configuración

ROL	RESPONSABILIDADES
AC (ADMINISTRADOR DE CONFIGURACIONES)	Asegurar que la políticas de la gestión de configuración se lleve a cabo Responsable de la creación y administración de ramas criticas Encargado de añadir etiquetas en la rama central y en las ramas de lanzamientos Controlar la unión de las ramas críticas. Asegurar la integridad y consistencia del producto.
EQUIPO	Ayudar a resolver conflictos surgidos cuando hay unión de ramas. Debe asegurar la calidad de su modulo. Llevar a cabo las políticas de Configuración propuestas por el AC Comunicar al administrador en caso de encontrar fallas.

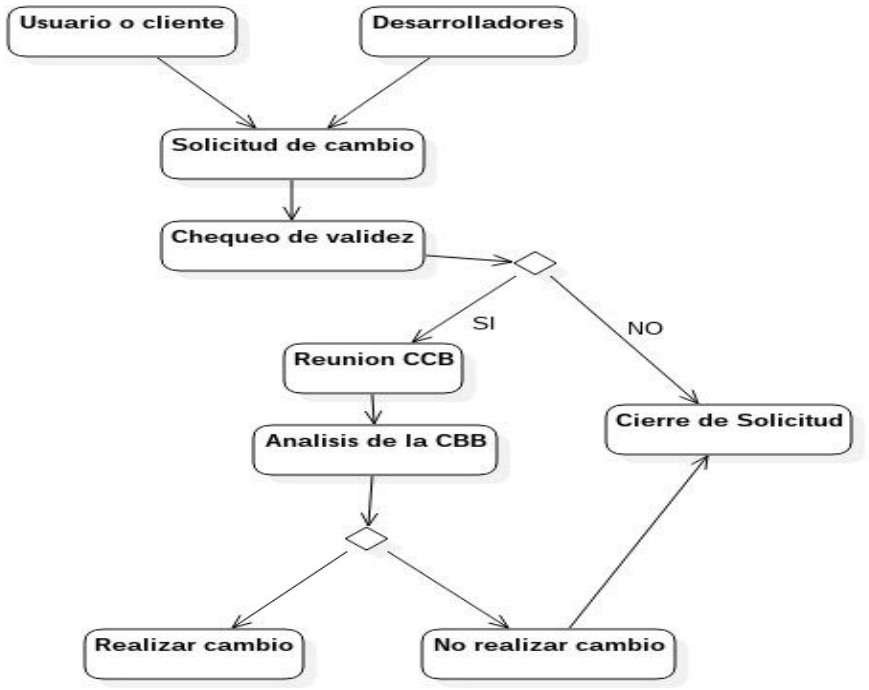
SECCION 3

Control de Cambios

Panorama

La idea general del control de cambios, es analizar la viabilidad y relación costo-beneficio de aplicar una modificación al programa en cuestión. Los cambios se pueden solicitar ya sea por una nueva funcionalidad exigida por un usuario, cliente o los mismos desarrolladores o el arreglo de bugs; esto va a ser una de las principales cuestiones a tener cuenta a la hora de analizar los cambios. Por lo tanto si la modificación solicitada es el arreglo de un bug que causa la disfuncionalidad del software, este será un cambio con mayor prioridad que otros. Además se tienen en cuenta otras cuestiones por ejemplo no se agregara una nueva funcionalidad al software si recientemente se realizó un lanzamiento mayor.

Procedimientos



El procedimiento a la hora de analizar un cambio es como lo muestra la figura. Primero alguien solicita una modificación, luego algún miembro del CCB chequea su validez (es decir que no sea imposible y que tenga el mínimo sentido), luego se efectúa una reunión del comité de control de cambios para analizar esta solicitud y por último se hace una devolución con la decisión tomada.

Comité del Control de Cambios

La CCB (Comité de Control de Cambios) por sus siglas en inglés, Change Control Board, es un comité directivo formado por un grupo de personas con los conocimientos técnicos suficientes para hacer un análisis objetivo y cuantitativo de la relación costo beneficio de implementar un cambio. Este grupo es el encargado de tomar la decisión final sobre que modificaciones tienen prioridad y cuales se van a implementar y cuáles no. Algunas cosas a tener en cuenta por la CCB antes de tomar una decisión son:

- Las consecuencias de no realizar el cambio.
- Los beneficios del mismo.
- El número de usuarios afectados.
- El costo de implementarlo.
- El ciclo de lanzamiento del producto.

Miembros

La siguiente tabla muestra los integrantes de las reuniones del comité técnico del control de cambios. En caso de no estar presente alguno se tomara la decisión sin esa persona, y esa persona tiene que acatar a la decisión que se tomó.

ROLES EN EL COMITE TECNICO DEL CONTROL DE CAMBIO	NOMBRE
DIRECTOR EN INGENIERIA	Fabian Wolfmann
COORDINADOR DE LANZAMIENTOS	Agustin Colazo
CM DEL PROYECTO GLOBAL	Nicolas Passaglia

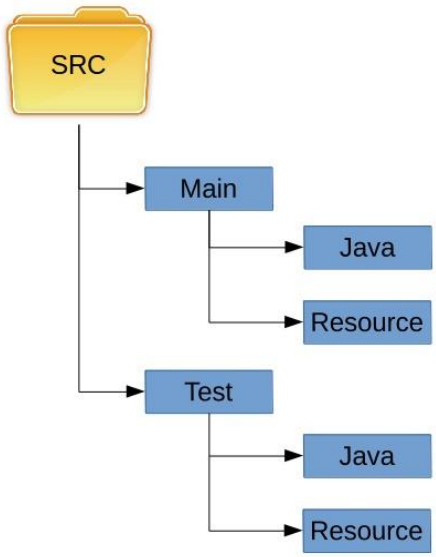
Frecuencia de las juntas

La frecuencia de reunión se va a ver afectadas por el número de cambios solicitados, es decir se juntaran siempre y cuando haya una modificación que analizar. Esto puede suceder todos los días.

SECCION 4

Esquema de directorios

A continuación se muestra la forma para implementar el esquema de directorios en el proyecto



En las subcarpetas se encuentran la carpeta main, donde se encuentra el código en java del programa y los recursos que este utilice. También se encuentra la subcarpeta de Test donde están los test a correr cuando sea necesario.

Política de nombramiento de archivos

Todos los archivos pertenecientes al proyecto, se incluirán en la carpeta SRC, es decir cada vez que se haga referencia a un archivo del proyecto será de la forma /SRC/SubCarpeta_Donde_Pertenece/Nombre_Archivo.Formato_Archivo. Cuando se trate de nombre de clase, interfaces y demás la primera letra será en mayúscula y el resto en minúsculas: Por ejemplo /SRC/Main/Java/Menu.java para hacer referencia al archivo que define la clase Menu del proyecto, por lo tanto en la carpeta Java se almacenaran los códigos fuente. Luego en la carpeta Resource de Main se guardará archivos todos aquellos necesarios para el proyecto, que no sean código fuente.

Con respecto a la sección de test, en la subcarpeta java estarán los códigos de las pruebas unitarias, cuyos nombres serán el código del archivo que hace las pruebas con la primera letra en mayúscula seguido de _UTest Ejemplo: SRC/Test/Java/Menu_UTest.java

Y en cuanto a los recursos de las pruebas, se encuentra toda la documentación especificando las funciones y los valores que devuelven, y además contiene todos los demás archivos necesarios para el funcionamiento de las pruebas unitarias (Imágenes, Sonidos, etc.)

SECCION 5

Equipos en Proyecto Final

Esta sección contiene los equipos que actúan en el Proyecto Final:

- **Equipo Scrum:** Los equipos Scrum son aquellos que se dedican a desarrollar nuevas funcionalidades utilizando algún proceso ágil. Los miembros van comprometiendo el código a la rama para que luego pueda ser unido en la misma rama de integración al final del sprint. Además a este tipo de equipo se le puede asignar tareas como el arreglo de errores y demás.

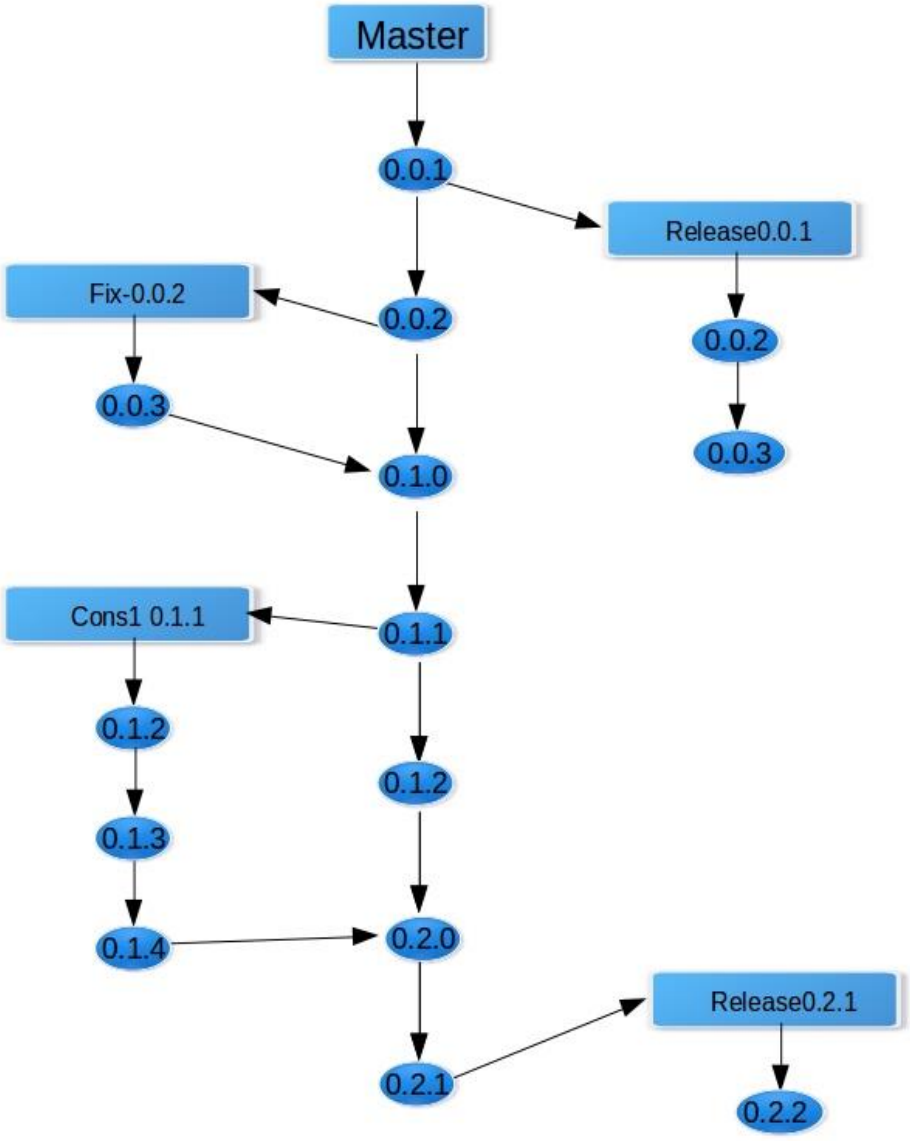
- **Equipo de manejo de lanzamientos:** El equipo de manejo de lanzamientos es el encargado de realizar todos los test necesarios sobre el producto para definir el lanzamiento final del mismo. Los errores arreglados por la RMT van a ser agregados a la rama donde se está desarrollando el lanzamiento.
- **Equipo de reacción rápida:** Este equipo se encarga de solucionar problemas identificados por clientes, estos problemas suelen necesitar una solución urgente por lo que no pueden esperar al siguiente lanzamiento, lo suelen resolver de la forma de instantánea. Los arreglos deben seguir los criterios definidos
- **Equipo de documentación de producto:** Son los encargados de realizar y mantener la documentación del proyecto que luego será entregada al cliente.

SECCION 6

En esta sección se describen los distintos elementos en la gestión del código. Cubre algunos aspectos del esquema de ramas, etiquetado, estrategia de unión de ramas y los niveles de calidad esperados del producto.

Administración del código fuente

El siguiente grafico muestra el esquema de ramas que se utilizara en el proyecto.



Categorización de ramas

La ramificación en la gestión de software es la duplicación de código para poder hacer un desarrollo paralelo en ambas ramas.

Los tipos de ramas que se van a utilizar serán:

- Rama central “master”: Esta es la rama principal donde se encuentra el código fuente del proyecto y todos los avances que se hicieron en el mismo. La rama central tendrá el nombre “master”.
- Ramas de desarrollo: Estas son las ramas donde las nuevas características/funcionalidades a desarrollar son codificadas. Las ramas de desarrollo se nombraran de la siguiente manera: D_NombredeRama
- Ramas de arreglo de errores: Las ramas de arreglo de errores son utilizadas cuando se identifican errores en el sistema, son utilizadas para solucionar estos fallos. Se nombran con la siguiente leyenda: “fix-NombreDeRama”
- Ramas de lanzamiento: Este tipo de ramas debe ser creado por el administrador de las configuraciones y son creadas cada vez que se lanza una nueva versión del producto. El nombramiento de las ramas de lanzamiento tendrá la siguiente leyenda: “Release X” donde X será una cifra que corresponde a la versión del producto que será lanzada.
  *Los lanzamientos de nuevas funcionalidades o sobre arreglo de errores, continúan en la misma rama que su predecesor.

La idea principal de este esquema de ramificación es para realizar los nuevos desarrollos y el arreglo de errores separado de la rama central. Esto se hace para mantener aislado el código que se está modificando del código ya desarrollado y estable que se encuentra en la rama central. El código en la rama central (master) es el código potencial para ser lanzado en la próxima versión del producto.

Las ramas de lanzamiento se aíslan de la rama principal de modo que el arreglo de bugs sobre estas versiones se lleve a cabo allí, y si se considera oportuno se podrá hacer una unión de la rama de lanzamiento a la rama central. También se puede utilizar las ramas centrales en caso de clientes particulares que tengan una versión vieja del producto y que estos hagan un pedido por nuevas funcionalidades o reporten fallos en el sistema.

Control de versionado

Se etiquetaran los archivos comprometidos si el funcionamiento de los mismos es comprobado por las pruebas unitarias. El nombramiento de etiquetas se utiliza para el control de versionado.

Ejemplo: PC.Cifra_Mayor.Cifra_Media.Cifra_Menor

Donde aparece cada cifra, ira un digito que representara la evolución del software en distintos niveles del desarrollo.

- Cifra Mayor: La cifra mayor corresponde al lanzamiento de una nueva versión del producto, donde este tendrá el agregado de varias funcionalidades nuevas que tendrán un gran impacto en el software. Esta cifra mayor se modificara cuando se haga una nueva rama de lanzamiento. La creación de estas ramas deben ser autorizadas por el coordinador de lanzamientos.
- Cifra Media: Las cifras medias se modifican cuando se hace una combinación entre ramas, cuando se agrega una nueva funcionalidad o se arreglan errores de software. El archivo ChangeNotes.txt se utiliza para llevar un registro de estos cambios.
- Cifra Menor: La cifra menor se modifica para llevar registro de la evolución de una rama de desarrollo o una rama de arreglo de errores, antes de que esta se una a la rama de la que partió.
  - En la rama master la cifra mayor tendrá el valor de “0” (cero) ya que esta versión no será lanzada directamente al mercado.

Archivos auxiliares

ChangeNotes.txt: Este archivo en ramas de lanzamiento o en la rama central cuando se agregan nuevas funcionalidades o arreglan fallos, en la unión con ramas de desarrollo o ramas de arreglo de errores. Esto se hace para tener un registro de los cambios que se van introduciendo en la rama. Los desarrolladores deben agregar una nota en la parte superior de este archivo sobre los cambios que están agregando y de ser necesario, una breve explicación sobre los cambios que realizaron.

Si dos ramas de lanzamiento se unen entre sí, o se une una rama de lanzamiento a la rama central (master) también se debe modificar el archivo ChangeNotes.txt para poder tener un registro de los cambios que se han hecho en la misma.

No es necesario usar estos archivos en las ramas de desarrollo, pero si es obligatorio utilizarlos para tener un registro de la evolución en las ramas de lanzamiento o en la rama central.

Ejemplo:
Versión: VX.X
Autor: <Nombre_del_Desarrollador>
Características Añadidas:
Característica 1
Característica 2: [EXPLICACION]
Errores Arreglados:
Error 1
Error 2: [EXPLICACION]
Comentarios:
[Información adicional]

Estrategia de unión de ramas

Cada vez que se vaya a agregar o modificar una funcionalidad, o a arreglar algún error en el sistema. Se creara una nueva rama a partir de la rama base (que puede ser la rama master o una rama de lanzamiento). Una vez desarrollada y probada la nueva rama, si pasa todas las pruebas, se procederá a fusionarla con la rama base. Es absolutamente obligatorio que para poder unir dos ramas es necesario que esta funcione correctamente, y el funcionamiento correcto de la misma es responsabilidad del desarrollador que combino ambas ramas. Dicho de otra manera, para la correcta evolución del proyecto, está prohibido fusionar dos ramas y que el funcionamiento de la resultante sea menor al funcionamiento anterior. Por lo tanto, el desarrollador tiene la obligación de que antes de hacer la unión entre dos ramas, se asegure que funcione correctamente.

Que significa que el funcionamiento deba ser igual o mayor, esto significa que si por ejemplo, la rama central funciona correctamente en 10 pruebas y se la va a combinar con una rama de desarrollo. El resultado debe pasar estas 10 pruebas y todas las pruebas que se le agreguen (por la nueva funcionalidad).

Ramas por cliente

En caso de que un cliente especifico haga un pedido por nuevas funcionalidades y el pedido sea aprobado por una jerarquía superior. Se creara una rama de lanzamiento específica para este cliente a partir de la rama que contiene la misma versión del software entregada a este cliente y se trabajara sobre esta rama.

Niveles de calidad

La rama principal y las ramas de lanzamiento deben cumplir con un determinado criterio de calidad ya que estos son las versiones potenciales del software a ser lanzadas al mercado, o entregadas al cliente. El equipo de desarrollo se encargara de la calidad del producto. Las ramas de desarrollo no han de cumplir requisitos de calidad, pero si se va a fusionar estas ramas con la rama central o una rama de lanzamiento, las primeras deben cumplir los mismos requisitos de calidad y estar terminadas antes de ser combinadas con las últimas.

Si una rama de desarrollo no logra el criterio de realizado o el criterio de la calidad, entonces bajo ningún contexto debe ser fusionada con la rama central o con una rama de lanzamiento.

Antes de la fecha de lanzamiento no se fusionara esta rama con ramas de desarrollo, a menos que esto sea considerado esencial para la supervivencia del proyecto.

El criterio de realizado o terminado corresponde cuando la nueva funcionalidad a desarrollar hace lo que se propuso en un principio que esta realizaría; o cuando el fallo que se iba a arreglar está resuelto, sin haber generado fallos en otras partes del sistema.

El criterio de calidad es cuando el código satisface todas las pruebas unitarias y de integración.

Las ramas de lanzamiento no son necesariamente lanzadas al mercado en su primera versión, se puede trabajar sobre estas modificando cuestiones de poca magnitud, preparándolas para ser lanzadas al mercado; buscando que satisfagan un criterio de calidad acorde a un producto que será entregado a los clientes.

Gestión de defectos

Si se encuentra un defecto en el proyecto, el mismo debe ser reportado por la herramienta para gestión de defectos de github. Se debe crear una entrada en la sección “issues” de github en donde se debe especificar información sobre el error encontrado. Luego, para solucionar el fallo se debe crear una nueva rama de arreglo de errores. Una vez solucionado el problema, se une esta rama a la rama de la que partió y se cierra la entrada en la sección “issues” de github para especificar que el mismo fue resuelto. Antes de cerrar la entrada, se debe dejar un comentario que especifique el nombre de la rama en que se soluciono el problema.

SECCION 7

Administración de Compilación de Software

En el proyecto se utilizaran compilación de integración continua que serán ejecutadas automáticamente por la herramienta Travis CI. Esta herramienta compilara el código y ejecutara todas las pruebas unitarias y de integración sobre el mismo, si el código comprometido satisface todas estas pruebas, entonces será agregado al repositorio común. Si no las satisface, el código no será añadido al repositorio. Esta herramienta también generara reportes sobre el resultado de las pruebas.

Por su lado, los desarrolladores también pueden compilar el código (de ser posible) en sus repositorios locales, para poder comprobar que los cambios hechos funcionen adecuadamente. Pero estos ejecutables no deben ser agregados al repositorio.

En las rama central y en las ramas de lanzamiento es obligatorio pasar por todas las pruebas y que el código compile. En las ramas de desarrollo esto no es necesario.

SECCION 8

Control de Lanzamientos

Habrá tres tipos de lanzamiento. El lanzamiento de una nueva versión de producto, este lanzamiento tendrá varias funcionalidades nuevas que significan un gran cambio para el producto. Cuando hay un lanzamiento de una nueva versión del producto se creara una nueva rama para tener registro de esta versión, en caso de que se solicite agregar nueva funcionalidad en la misma, o arreglar errores de software.

Luego, habrá lanzamientos mayores y menores. Los lanzamientos mayores corresponderán al agregado de nuevas funcionalidades mínimas sobre versiones anteriores del software, estos lanzamientos generalmente serán sin costo. Los lanzamientos menores serán entregados al cliente de manera gratuita y corresponden al arreglo de fallos en el sistema.

El coordinador de lanzamientos es quien determinara que tipo de lanzamiento será determinada versión del sistema, y cuando se harán estos.

El formato para los distintos lanzamientos será de la siguiente manera: Nombre_Proyecto.Version.Lanzamiento_Mayor.Lanzamiento_Menor.

Las cifras serán modificadas dependiendo del lanzamiento del que se trate.