Grupo VLC.

FCEFyNApp.

Configuration Management Plan

(CM Plan)

Autores: López Gastón, Vignolles Iván, Lamberti Germán.

Versión del documento: 2.0.0

Materia: Ingeniería del Software.

Profesor: Nonino, Julián.

Facultad de Ciencias Exactas, Físicas y Naturales.

Universidad Nacional de Córdoba.

Año 2017.

Historial de revisiones.

Versión	Fecha	Resumen de cambios.	Autor/es
1.0.0	8/04/2017	Documento inicial.	López Gastón, Vignolles Iván, Lamberti Germán.
2.0.0	6/05/2017	Nombre de la app. Glosario. Administración de cambios, de builds y de releases. Estrategias de fusión de archivos. Roles y responsabilidades de los integrantes del equipo.	López Gastón.

Índice.

1 INTRODUCCIÓN…… 4
1.1 PROPÓSITO …… 4
1.2 GLOSARIO..…….. 5
1.3 HERRAMIENTAS DE ADMINISTRACIÓN Y/O CONTROL DE LAS CONFIGURACIONES ……..……... 7

2 ADMINISTRACIÓN DEL MANEJO DE LAS CONFIGURACIONES…….....7
2.1 ROLES Y RESPONSABILIDADES..……....7

3 ADMINISTRACIÓN DE CAMBIOS…….…….... 9
3.1 SOLICITUD DE CAMBIO…….. 9
3.2 CCB (COMITÉ DE CONTROL DE CAMBIOS)……..... 9
3.3 PROCESO DE CONTROL DE CAMBIOS…….11

4 IDENTIFICACIÓN DE LAS CONFIGURACIONES……..11

5 EQUIPOS DE Students Center App……..…….. 12

6 ADMINISTRACIÓN DEL CÓDIGO FUENTE…….13
6.1 ESQUEMA DE RAMAS…….13
6.2 DEFINICIÓN DE ETIQUETAS…… 14
6.3 ESTRATEGIA DE FUSIÓN DE ARCHIVOS…….15
6.4 RAMAS POR CLIENTE…….16

7 ADMINISTRACIÓN DE BUILDS…… 16

8 ADMINISTRACIÓN DE RELEASES……...16

9 BACKUP……...17

CONFIGURATION MANAGEMENT PLAN.

1 INTRODUCCIÓN

Este documento describe el plan de manejo, gestión o administración de las configuraciones del proyecto final de la materia Ingeniería de Software de la Facultad de Ciencias Exactas, Físicas y Naturales (Universidad Nacional de Córdoba). El proceso de administración de las configuraciones asegura el control de las distintas versiones de los documentos y proyectos entregables, permite definir y visualizar las responsabilidades y roles del equipo que desarrolla dicho proyecto y posibilita realizar cambios en este último de acuerdo a un proceso sistemático conocido por todo el equipo, además de que permite un seguimiento y control del tiempo y de los recursos utilizados y generados durante el desarrollo.

1.1 PROPÓSITO

El propósito de este documento es el de establecer los elementos necesarios para administrar los códigos fuentes, archivos y documentos que son elaborados por el equipo del proyecto. Entre otros propósitos se encuentran:

- Coordinar el uso y actualización de los distintos módulos que forman parte del proyecto.
- Asegurar que todos los integrantes del equipo estén trabajando en las mismas versiones de cambios y/o modificaciones.
- Controlar que ninguno de estos cambios se pierda.
- Definir las herramientas que se utilizarán para este proceso de administración y/o control de las configuraciones.
- Determinar cómo está conformado el equipo de trabajo y definir los roles y responsabilidades de cada uno de los integrantes en las distintas partes y etapas del proyecto.
- Definir un proceso sistemático que se debe seguir para poder realizar cambios en el proyecto.
- Definir las personas que deberán aprobar el cambio solicitado para el proyecto.
- Determinar y definir las nomenclaturas que se colocarán para el proceso de identificación de las configuraciones.

- Determinar el proceso de administración del código fuente, la forma en que se definirán las etiquetas, el esquema de ramas y la estrategia de fusión de archivos.
- Definir el proceso y las herramientas para la administración de builds.
- Definir cómo se efectuará un “backup” del proyecto.
- Definir formato y forma de entrega de releases, así como también la administración de los mismos.

1.2 GLOSARIO

Los siguientes términos, debido a su reiterada mención en el documento, es necesario definirlos para poder entender su significado:

Acrónimo o palabra.	**Significado.**
CM	Configuration Management. (Administración de las configuraciones).
Repositorio	Lugar donde se almacenan los ítems bajo control de configuración y sus versiones (es decir, datos actualizados e históricos), a menudo en un servidor. Puede ser un sistema de archivos en un disco duro, un banco de datos, etc.
Versión	Toda evolución de un ítem bajo control de configuración.
Release	Lanzamiento del producto, puesta en venta.
Cambio	Modificación específica de un documento bajo control de configuraciones.

CCB (Change Control Board o Comité de Control de Cambios).	Grupo de personas responsables de evaluar y aprobar determinados cambios sobre un ítem de configuración o proyecto de software.
Build	Generar, compilar y/o ejecutar un sistema.
Backup	Copia de los datos originales que se realiza con el fin de disponer de un medio para recuperarlos en caso de su pérdida.
Ejecutable.	En este documento, hace referencia al archivo con extensión "*.apk*" que es el que se sube en Google Play Developer Console.
Ítem bajo control de configuración.	Cualquier elemento que esté bajo control de configuración en el repositorio.
Commit	Acción de guardar los archivos en el repositorio remoto (si es que se realiza esta acción desde el sitio web de GitHub, sino se debe realizar un *push*).
Push	Acción de guardar los archivos en el repositorio remoto (se utiliza esta acción fuera del sitio web de GitHub, cuando los cambios en los archivos se realizan en un ambiente privado de trabajo).
Etiqueta	Identificador pegado a alguna versión de determinado ítem bajo control de configuración.

Tabla 1. Acrónimos/Glosario.

1.3 HERRAMIENTAS DE ADMINISTRACIÓN Y/O CONTROL DE LAS CONFIGURACIONES.

Las herramientas denominadas a continuación serán utilizadas para facilitar el desarrollo del proyecto y asegurar una correcta integración del trabajo de los desarrolladores del equipo.

- Herramienta de control de versiones: se utilizará la herramienta GitHub. Se creará un repositorio en el servidor web de dicha herramienta, al cual van a tener acceso todos los integrantes del equipo como “colaboradores”. El nombre del repositorio es: GastonLopez / 2017-UNC-IngSoft-VLC. El link de dicho repositorio es: https://github.com/GastonLopez/2017-UNC-IngSoft-VLC.

- Herramienta de integración continua: se utilizará la herramienta Travis CI conjuntamente con Gradle (automatización de builds, sistema de automatización de compilación de código abierto). A medida que los desarrolladores realicen modificaciones en algún ítem de configuración, la herramienta Travis CI correrá los test automáticamente, corroborando que no se hayan introducido errores. En caso de que se detectaran, dicha herramienta se encargará de notificar a todos los desarrolladores. El nombre del repositorio es GastonLopez / 2017-UNC-IngSoft-VLC. El link se detalla a continuación: https://travis-ci.org/GastonLopez/2017-UNC-IngSoft-VLC

- Herramienta para gestión o control de defectos: se utilizará la herramienta Issues/GitHub para poder estar notificados y poder seguir las tareas, mejoras, errores e inconvenientes que van surgiendo en la confección del proyecto. El nombre del repositorio es GastonLopez / 2017-UNC-IngSoft-VLC. El link se detalla a continuación: https://github.com/GastonLopez/2017-UNC-IngSoft-VLC/issues.

2 ADMINISTRACIÓN DEL MANEJO DE LAS CONFIGURACIONES

2.1 ROLES Y RESPONSABILIDADES

Las actividades en cuanto a la administración y/o gestión de las configuraciones serán llevadas a cabo y coordinadas por los tres integrantes del equipo, debido a que todos están en el mismo nivel jerárquico (las responsabilidades en juego serán compartidas por los 3 integrantes). Existen determinadas funciones específicas que se detallan a continuación:

Roles específicos.	**Integrante.**
- Realizar, controlar y estar a cargo del backup del proyecto. - Analizar y controlar la información y los archivos subidos al repositorio. - Controlar y coordinar esfuerzos para la solución de los errores que aparecen en la integración continua.	López, Gastón.
- Definir y controlar estrategias de fusión de archivos. - Controlar la correcta identificación de archivos y configuraciones, definiciones de etiquetas y ramas.	Vignolles, Iván.
- Supervisar que el equipo de trabajo utilice el repositorio correspondiente en GitHub durante el desarrollo del proyecto. - Hacer respetar los lineamientos establecidos en este documento que deben seguir todos los procesos de solicitud y aprobación de cambios.	Lamberti, Germán.
- Controlar que el producto posea un correcto funcionamiento. - Controlar que el producto cumpla con los requerimientos establecidos.	Todo el equipo.

Tabla 2. Roles específicos.

Con el fin de lograr un proyecto válido, que cumpla con los requerimientos y que funcione correctamente, los tres integrantes serán responsables de cumplir con sus funciones específicas.

3 ADMINISTRACIÓN DE CAMBIOS

Los cambios pueden proceder tanto a la mejora del producto como a la corrección de errores.

El procedimiento para efectuar un cambio deberá comenzar con el comunicado y documentación de la solicitud de cambio. El mismo será evaluado por la totalidad de los integrantes del equipo. La votación determina si la solicitud se posterga, se rechaza o se acepta e implementa.

3.1 SOLICITUD DE CAMBIOS

Pasos necesarios para efectuar la solicitud de cambio/s:

1. Definir qué parte o sector del software se desea modificar.
2. Describir detalladamente los cambios a realizar y proponer títulos o nombres para esos cambios.
3. Explicar por qué se solicita realizar dichos cambios.
4. Proponer plazos y fechas límites para la implementación.
5. Una vez cumplido los pasos anteriores y habiendo escrito un documento con la información solicitada, se presentará el mismo al integrante del equipo encargado de hacer cumplir los lineamientos establecidos para los procesos de cambios (ver sección 2.1).
6. El encargado que figura en la sección 2.1 deberá someter dicha petición al control y evaluación de cambios correspondientes.

3.2 CCB (COMITÉ DE CONTROL DE CAMBIOS)

La CCB estará integrada por los tres integrantes de VLC. En dicha junta se analizarán, de acuerdo a los cambios propuestos, cómo se verán afectados los requerimientos del proyecto y se determinarán los riesgos, costos y ventajas que conllevan dichos cambios. La decisión final se tomará en base a una votación en la que participarán la totalidad de los integrantes. Para que un cambio se apruebe necesita, como mínimo, un voto positivo de las dos terceras partes de la junta.

Integrantes del CCB:

Integrantes.	**Función en la CCB.**
Todo el equipo.	-Planear, identificar, analizar y proporcionar las bases para poder hacer cambios en el desarrollo y en la implementación de los requerimientos. Determinar cómo se verán afectados dichos requerimientos con los cambios solicitados. - Aprobación, rechazo o postergación de la solicitud de cambio por votación.
Lamberti Germán.	- Reportar las solicitudes de cambios. - Reportar cambios en la duración del proyecto y en el planeamiento de recursos y costos. - Dar inicio al Comité de Control de Cambios. - Es el encargado de documentar el resultado del CCB.
López Gastón.	Administrador de riesgos y de costos (que conllevará la aprobación de los cambios).
Vignolles Iván.	Administrador de desarrolladores (se encargará de realizar aportes y estimaciones más reales en cuanto al desarrollo de la aplicación y los módulos que se verán afectados si se aprueban los cambios).

Tabla 3. Funciones en la CCB.

Reuniones: se requiere de una comunicación rápida y fluida entre los tres integrantes del Comité. Las reuniones se realizarán cuando exista alguna petición de cambio y la comunicación se realizará principalmente a través de un grupo de la app WhatsApp. En caso de tratarse de un cambio muy complejo, se pasará a una reunión en las instalaciones de la Facultad de Ciencias Exactas, Físicas y Naturales, con hora y fecha determinada por votación.

3.3 PROCESO DE CONTROL DE CAMBIOS

Una vez presentada la propuesta correctamente formulada y documentada (de acuerdo a la sección 3.1) dicha propuesta pasará al siguiente proceso para su evaluación, el cual consistirá en:

a) Definir si la información recibida es suficiente para la evaluación global del cambio. En caso de no ser suficiente, el solicitante del cambio deberá agregar más información a la propuesta o descartarla.
b) Llamar a CCB.
c) Los pasos “a” y “b” deberán ser realizados por la persona encargada de llamar a la CCB según sección 2.1.
d) En la reunión del Comité, se analizarán y determinarán los requerimientos afectados, módulos, tests y partes de los códigos fuente que se deberían modificar y los riesgos y costos que ello conllevaría.
e) Se realiza la votación en la Junta. Todo esto deberá estar debidamente documentado y almacenado por la persona que debe realizar los pasos **a** y **b**.
f) Si es aprobada la solicitud de cambio, se pasará a decidir por la CCB la implementación del cambio y los plazos que se definirán para ello. Todo esto deberá estar debidamente documentado por la persona que debe realizar los pasos **a** y **b**, la cual deberá almacenar dicho documento en la carpeta Cambios*Aprobados* en el repositorio de GitHub.
g) Si se posterga la solicitud de cambio, se deberá establecer fecha o condición para la nueva reunión de la Junta. Se deberán almacenar los documentos correspondientes al cambio en la carpeta *CambiosPostergados* en el repositorio de GitHub.
h) Si se rechaza la solicitud de cambio, se deberán almacenar los documentos correspondientes en la carpeta *CambiosPendientes* en el repositorio de GitHub.

4 IDENTIFICACIÓN DE LAS CONFIGURACIONES

El proyecto será guardado bajo el directorio GastonLopez / 2017-UNC-IngSoft-VLC. (Repositorio de GitHub).

En dicho repositorio se encuentran las siguientes carpetas:

- *src:* guarda el contenido de la aplicación (código fuente) y los tests.

- *docs:* guarda datos y archivos importantes del proyecto como ser el plan de CM, diagramas (en el subdirectorio "Diagramas"), imágenes, documento de requerimientos y archivo .apk (subdirectorio “ejecutables”).

- *bin:* guarda documentos compilados del proyecto. Además esta carpeta contiene archivos para el uso de integración continua, archivos .txt, etc., como son: travis.yml, README.md, entre otros.

- *Cambios:* posee tres subdirectorios que son CambiosPostergados, CambiosPendientes y CambiosAprobados (ver sección 3.3). Todos los documentos relacionados a las solicitudes de cambios y a las decisiones (y sus argumentos) de la CCB se almacenan en alguna de estas tres carpetas.

A continuación se muestra la organización del directorio del repositorio de GitHub: https://github.com/GastonLopez/2017-UNC-IngSoft-VLC/blob/master/docs/Imagenes/DirectorioYSignificado.png

Directorios	Significado
src	Códigos fuente del proyecto y tests.
bin	Uso de integración continua, archivos compilados del proyecto, README.
docs	Documentos asociados al proyecto.
docs/Imágenes	Imágenes asociadas al proyecto.
docs/Diagramas	Diagramas asociados al proyecto.
docs/Ejecutables	Ejecutables asociados al software.
Cambios	
Cambios/CambiosPendientes	Cambios rechazados por la CCB.
Cambios/CambiosPostergados	Cambios postergados por la CCB.
Cambios/CambiosAprobados	Cambios aprobados por la CCB.

Tabla 4. Directorio y significado en el repositorio de GitHub.

5 EQUIPOS DE FCEFyNApp

En total el equipo de VLC se compone de tres integrantes. Estos se pueden unir de distintas formas, conformado distintos equipos que se detallan a continuación:

Equipo Scrum: encargado de desarrollar nuevas funcionalidades de manera ágil y rápida. Actualiza el código frecuentemente, evitando problemas a la hora de unir las partes desarrolladas por cada uno. También pueden estar encargados de corregir bugs en el programa.

Equipo de administración de etiquetas: serán los encargados de establecer y controlar el cumplimiento de las normas de etiquetado.

Equipos de rápida reacción: se encargan de solucionar los errores urgentes (por razones de tiempo, necesitan ser solucionados).

Equipo de documentación: son los encargados de crear, modificar y mantener actualizada la documentación del producto que será entregada al cliente, la cual le permite entender al mismo las funcionalidades del proyecto, sus características y su implementación. Por ejemplo: archivos README, guía del usuario, manual de ayuda, respuestas a preguntas frecuentes, etc. También serán los encargados de mantener actualizada la documentación del proyecto, como ser este plan.

Equipo de nuevos desarrollos: el objetivo de este equipo será el de descubrir, identificar, analizar e implementar nuevos módulos que se pueden integrar al proyecto.

Equipo de control del repositorio: su función será la de corroborar que se esté utilizando el repositorio, la de identificar los archivos que se van subiendo y los cambios que se introducen en ellos, verificar que los nombres de versiones, ramas y archivos sean los adecuados y controlar que dichos archivos se coloquen en los directorios correspondientes.

6 ADMINISTRACIÓN DEL CÓDIGO FUENTE

En esta sección se describen el esquema de ramas, el etiquetado, la estrategia de fusión de archivos y el cumplimiento de los niveles de calidad para el producto final.

6.1 ESQUEMA DE RAMAS.

Posibles ramas a utilizarse en este proyecto:

Rama Master: es la rama principal. La mayoría de las fusiones de archivos se realizarán sobre esta rama. En ella se incluirá el desarrollo del software sin errores que cumpla con algunos o con todos los requerimientos. Las versiones a lanzarse se colocarán en esta rama, siempre y cuando no aparezcan clientes con nuevos requerimientos ni el cliente inicial haga un cambio de ellos y la CCB lo apruebe.

Rama de desarrollo: rama donde se codifica el desarrollo de nuevas características y funcionalidades acordes a los requerimientos que debe cumplir el proyecto. Generalmente estas ramas son utilizadas por los equipos Scrum.

Rama de función: la creación de la rama tiene lugar debido a la modificación de una o varias funcionalidades y sus implementaciones. (Sirven para cuando se necesita ir realizando correcciones parciales por los errores o defectos que se encuentran en las corridas de prueba).

Rama de cliente: un cliente puede solicitar un cambio que, si es aprobado por la CCB, origina una nueva rama.

Rama de modificación de la documentación: Este tipo de rama será utilizada para realizar modificaciones en los documentos asociados al proyecto, tales como requerimientos, diagramas, etc. La identificación de la rama deberá corresponder al documento en cuestión.

La imagen correspondiente al esquema de ramas se incluirá en versiones posteriores del documento debido a la falta de información presente en esta etapa del desarrollo del proyecto. Por el momento, se podría decir que en principio se tendría una *rama master* y una *rama de desarrollo*. Ésta última se utilizaría para la implementación de los requerimientos a cumplir. A medida que se vayan completando dichas implementaciones, se realizarán las respectivas fusiones sobre la *rama master*.

6.2 DEFINICIÓN DE ETIQUETAS

Se etiquetará un *commit* o *push* sólo si un código cuenta con correcta sintaxis y funciona correctamente. El etiquetado debe ser autorizado y revisado por el equipo de administración de etiquetas. También deberán ser respetadas las reglas que aquí se establecen para el nombramiento de dicha etiqueta. Estas serán nominadas de la siguiente forma:

Num1.Num2.Num3

- Primer dígito (Num1): este release indicará un gran cambio en la funcionalidad del proyecto (agregando al menos dos características nuevas). Cada uno de estos releases deberá contar con autorización de los tres

integrantes del grupo VLC, o en su defecto, por la mayoría, ya que esta será una versión nueva por la cual los clientes deberán pagar.

- Segundo dígito (Num2): este dígito indica que solamente una funcionalidad es lo que se difiere en cuanto a la anterior versión (cambios intermedios). Lo que normalmente ocurre es que se espera que aparezca algún otro cambio en funcionalidades u algún otro release para poder entregar una versión más completa al cliente. Por otra parte, si no ocurre nada de lo anterior, se determina por votación del equipo si esta versión va a estar disponible para los mencionados clientes.
- Tercer dígito (Num3): incluirá correcciones de bugs y modificaciones mínimas en el diseño de la interfaz que no afecten el funcionamiento del sistema (cambios menores). Formarán parte de actualizaciones gratuitas de la versión anterior ya adquirida por el cliente.

6.3 ESTRATEGIA DE FUSIÓN DE ARCHIVOS

Se utilizará una política de trabajo que permitirá que integrantes del equipo VLC trabajen en simultáneo sobre el mismo proyecto, sin molestarse y de una manera rápida y eficiente, posibilitando la ampliación y creación de funcionalidades. Para poder llevar a cabo esto, se utilizarán varias ramas. Éstas deberán unirse en algún momento con la rama Master o de integración. El problema muchas veces surge cuando se debe realizar el merge (fusión) entre distintos ítems de configuración y aparecen conflictos que deben ser resueltos. Es por esto que se establecen las siguientes reglas:

- La fusión debe ser aprobada por la totalidad de los integrantes, o en su defecto, por las dos terceras partes del equipo VLC.

- Se especificará la etiqueta de cierre (versión a la cual se la quiere fusionar) y la etiqueta inicial (versión donde nace la rama).

- Una vez terminada la fusión, será revisada por el encargado según sección 2.1 para confirmar y controlar los cambios realizados en la rama de integración (Master).

- Si los conflictos persisten entrará el equipo de Scrum (puede estar conformado por uno o más integrantes del grupo), el cual puede decidir aislar algunos ítems en ambas ramas para lograr la correcta fusión. Todo esto

deberá documentarse en un archivo denominado “EstadoDeLasRamas.txt”, ubicado en el directorio *docs*.

- En el mismo archivo, se detallarán los estados y resultados de las fusiones de las ramas y los motivos de la creación de las mismas.

6.4 RAMAS POR CLIENTE

Como se mencionó en la sección 6.1, para los cambios específicos de los clientes se crearán ramas, las cuales irán evolucionando y adecuándose, por cuenta propia, de acuerdo a los requerimientos de dichos clientes.

7 ADMINISTRACIÓN DE BUILDS

En el proyecto se utilizarán builds de integración continua. Se usará la herramienta Travis CI sincronizada con GitHub. (Se utilizará también la herramienta Gradle). Estos builds se ejecutan cada vez que un desarrollador hace un *commit* o *push* en el repositorio de GitHub, asegurando que éste no rompa el código por los cambios que efectuó. Fuera ese el caso, la herramienta deja asentada quien fue el desarrollador que lo hizo y además indica en donde está el error, notificando por mail sobre las fallas a todo el equipo. A su vez, Travis CI se encargará de ejecutar los tests y de efectuar los reportes correspondientes de acuerdo a los resultados de dichos tests. Los desarrolladores, a su vez, efectuarán builds a nivel local en forma informal para asegurarse de que sus implementaciones sean correctas. Estos últimos no se enviarán al repositorio.

8 ADMINISTRACIÓN DE RELEASES

El producto a entregar será consistente en una carpeta comprimida en formato “*.zip”* con contraseña con un directorio que incluirá documentos del usuario (FCEFyN) (contratos, políticas de privacidad, documento de requerimientos, etc.) y, por otro lado, un archivo “.apk” listo para subirse en “Google Play Developer

Console", donde el cliente (FCEFyN) deberá tener una cuenta registrada. Dicho archivo ".apk" estará acompañado por datos y archivos de configuración (relacionados con la utilización de los servidores). Cabe destacar que debido a que la empresa Google exige una política de privacidad cuando se requieren utilizar determinados permisos y accesos a información personal, el usuario (FCEFyN en este caso) deberá alojar en algún servidor el archivo que contiene dichas políticas de privacidad de la app debido a que se necesitará su link para poder cumplir con las condiciones que exige Google.

Por otra parte, en cuanto a la administración de los lanzamientos, se realizarán compilaciones y construcciones de sistemas sobre la rama de lanzamiento (Master o de integración), teniendo en cuenta el cliente del cual se trata o de los requerimientos que éste impone (la rama puede variar dependiendo del cliente o de los requerimientos que impone). A medida que se van realizando los distintos tests con un desarrollo incremental iterativo, se irán corrigiendo los diferentes defectos y errores que van surgiendo. Cabe destacar que se deberán identificar correctamente los sistemas que se construyen en cada ciclo (ver sección 6.2). El producto, una vez que pasa todas las pruebas a las cuales se lo somete y que estarán efectuadas de acuerdo al Documento de Requerimientos, será otorgado al cliente.

Se deberán detallar en el documento "Releases.txt" los distintos releases efectuados y sus respectivos clientes. Dichos releases serán nombrados de acuerdo a las normas de etiquetado dadas en la sección 6.2, anteponiendo o indicando (de alguna forma) la palabra clave *Release*.

9 BACKUP

El proyecto se encontrará almacenado y alojado en los servidores de GitHub (ver Sección 1.3). Sin embargo, por cuestiones de seguridad del trabajo efectuado, se realizará una copia del repositorio en el ordenador personal y en un pendrive de uno de los integrantes del grupo (ver Sección 2.1) con una periodicidad variable de acuerdo a los avances y cantidad de cambios que se vayan efectuando a lo largo del desarrollo del proyecto. En primer lugar, se realizaría el backup cada 2 días de trabajo y de cambios en el repositorio.