**Administración de las configuraciones**

**Introducción**

Este plan de administración de configuraciones, corresponde al trabajo final de la materia Ingeniería de Software. En el mismo se buscará describir las Gestiones tanto de cambio, versiones, release y las herramientas utilizadas para la construcción del ejecutable del mismo.

**Propósito:**

El propósito de este documento, es controlar las políticas, los procesos y las herramientas para administrar el software creado por el grupo.

* Mantener un control de cambio de manera que este, realizado por un desarrollador, no afecte en el código de otro.
* Controlar que todos los cambios sean guardados.
* Informar al que lo lea, el estado del proyecto.

**Glosario**

Los términos a continuación, serán utilizados frecuentemente, por lo que los dejaremos definidos, para una mejor comprensión.

**Repositorio:** Espacio de trabajo, en el cual se guardan todos los documentos, códigos fuente y actualizaciones del proyecto.

**Rama (Branch):** Es el camino independiente del desarrollo del proyecto, de manera que cada desarrollador, realice los cambios en el mismo, de forma individual.

**SCI (System Configuration Item):** El item de configuración, es cualquier aspecto asociado con el proyecto.

**CCB (Change Control Board):** Grupo que se encarga de evaluar y aprobar los cambios que se realizarán al proyecto.

**Herramientas utilizadas**

En esta parte del documento, se describirán las aplicaciones utilizadas para realizar el desarrollo del proyecto, de manera que este sea más sencillo y el resultado final sea más profesional.

* **Lenguaje de Programación:** Java

[**https://www.java.com**](https://www.java.com)

* **Entorno de Desarrollo:** Android Studio

[**https://developer.android.com/studio/index.html?hl=es-419**](https://developer.android.com/studio/index.html?hl=es-419)

* **Software de Manejo de Versiones:** Git

[**https://git-scm.com/**](https://git-scm.com/)

* **Sistema de Control de Versiones**: GitHub

<https://github.com/LuisLenta/EditorDeImagenes>

* **Sistema de Seguimiento de Errores:** Git Issues https://github.com/LuisLenta/EditorDeImagenes/Issues

**Roles y responsabilidades**

Las actividades serán coordinadas por tres integrantes. Cada integrante tendrá el mismo nivel jerárquico, pero los roles dentro del proyecto, serán distintas, las cuales se detallan a continuación:

**Administrador de la Configuración:**

El encargado del mismo, verificará que el equipo guarde los cambios realizados dentro del repositorio predefinido.

Encargado**:** (Gon, Alexander Mateo)

Email**:** alexg\_13\_92@hotmail.com

**Administrador de Pruebas:**

Se encargará de controlar que el sistema funcione de manera adecuada y cumpla con los requerimientos especificados por el usuario.

Encargado**:** (Orecchini Alem, Stefano Mauricio)

Email:stefano.itr.unc@gmail.com

**Administrador de construcción y release:**

Realizará la construcción del ejecutable para la entrega al cliente, además de verificar que el release cumpla con las opciones o requerimientos especificados durante la etapa de desarrollo.

Encargado:(Lenta, Luis Alejandro).

Email:[luislenta@gmail.com](mailto:luislenta@gmail.com)

**Administración de cambios:**

Esta sección del documento, se refiere al análisis de los costos y beneficios de los cambios propuestos, aprobando aquellos que lo ameritan e indagando cual o cuales de los componentes del sistema se modificarán.

Se entiende como cambio, todo lo que afecte la línea base del sistema. Este podría ser el reporte de un bug, en el que se describen sus síntomas, o una petición para agregar alguna funcionalidad al sistema.

Esta administración, será realizada por el CCB, el cual estará conformado por:

* **Gerente de Proyecto:** Este miembro, será el responsable de monitorear el desarrollo del proyecto. (Gon, Alexander Mateo)
* **Encargado de Documentación:** Será el responsable de documentar la información de los cambios. (Orecchini Alem, Stefano Mauricio)
* **Encargado de Aprobación:** Este verificará si es viable el cambio, tanto a nivel de software como empresarial. (Gon, Alexander Mateo)
* **Desarrollador:** Responsable de realizar los cambios correspondientes en el proyecto y los entregables. (Lenta, Luis Alejandro)
* **Testing:** Encargado de revisar los resultados finales de los cambios realizados, antes de entregar el release. (Orecchini Alem, Stefano Mauricio).

Las reuniones para analizar los cambios que se vayan presentando, a medida que se desarrolle el proyecto, serán semanales y en las cuales deberán estar los tres integrantes del grupo. Los días y horarios de las mismas, serán a convenir vía WhatsApp.

**Solicitud de Cambio:**

Para realizar esto, se deberá llenar un formulario, el cual constará con los siguientes ítems a completar:

* **Fecha de la solicitud:** Se deberá completar con la fecha en que el cambio es solicitado.
* **Cambio solicitado:** Especificación del cambio que se desea realizar.
* **Solicitante del cambio:** Identificación personal del mismo.
* **Analizador del cambio:** Identificación personal del mismo.
* **Fecha de Análisis:** Fecha en que se evalúa el mismo.
* **Componentes Afectados:** Partes del software que deberán ser modificadas.
* **Valoración del Cambio:** Breve reseña del analizador del cambio.
* **Prioridad del mismo:** Importancia que el CCB le da al cambio.
* **Decisión Final:** Aprobación o rechazo del mismo.
* **Fecha de Decisión de la CCB:** Fecha en que se aprobó o rechazó el cambio.
* **Implementador del Cambio:** Identificación personal del desarrollador que lleva a cabo el cambio.

**Actividades del CCB:**

Una vez que se completó el formulario de cambio mencionado anteriormente, este llega al CCB, el cual lo evalúa y define una respuesta.

Los pasos a seguir para evaluar el mismo son:

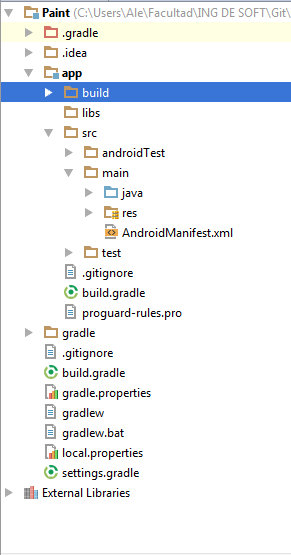
1. Verificar que los detalles proporcionados en el formulario, sean suficientes como para poder realizarlo.
2. Analizar las consecuencias de no realizar el cambio. Como así también los beneficios del mismo.
3. Los costos de realizar el cambio: Si el cambio afecta muchos componentes del sistema, y/o tarda mucho tiempo en implementarse.
4. Convocar a la CCB para el análisis económico y evaluación integra del cambio requerido.

En base al análisis de los puntos mencionados anteriormente, la CCB tomará la decisión final. Si el cambio es aceptado, se elevará el pedido a los desarrolladores para su correspondiente implementación, caso contrario, se archivará.

**Esquema de directorios y propósito de cada uno**

|  |  |
| --- | --- |
| **Directorio** | **Contenido** |
| **.\app\src\main** | Contiene las clases principales del proyecto |
| **.\app\src\test** | Contiene las clases de test usadas en el proyecto. |
| **.\graddle\** | Contiene el wrapper de Gradle. |
| **.\app\build\** | Contiene información necesaria para la compilación del proyecto |
| **.\docs\** | Contiene documentación y diagramas del proyecto. |

A continuación se muestra una imagen de la organización del directorio



**Normas de etiquetado y nombramiento de los archivos**

Esta se realizara mediante tres dígitos, separados por cada uno por un punto, los cuales cumplirán distintos roles que serán detallados a continuación:

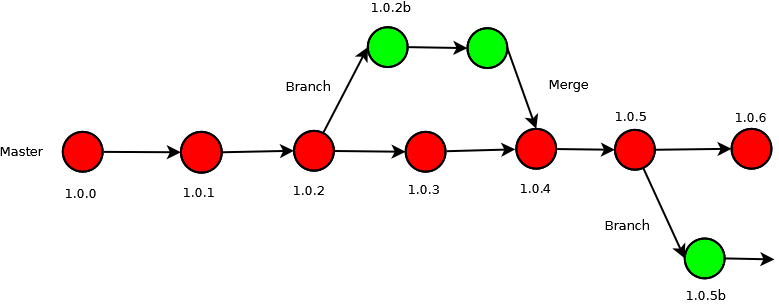
* Primer Dígito: Representa un gran cambio en la funcionalidad del sistema. Agregando características distintivas.
* Segundo Dígito: Representa pequeños cambios en la funcionalidad del sistema.
* Tercer Dígito: Este será utilizado para los bugs, es decir cada vez que se solucionen los mismos, actualizaremos este dígito.

En el caso de la creación de un branch, colocaremos la letra b al final de la numeracion.

**Plan del esquema de ramas a usar (y en uso)**

Generalmente trabajaremos sobre el master (línea base) del proyecto, pero en ciertas ocasiones será necesario crear una línea de desarrollo paralelo, para esto usaremos los branch(ramas). Se entiende por branch a una línea distinta de la línea principal o base.

Los branch se crean en momentos que dos o más desarrolladores necesitan trabajar paralelamente en el proyecto sin molestarse entre sí o cuando surge algún error crítico y se debe trabajar de forma inmediata sobre la versión que hay en producción y sin pasar toda la funcionalidad de las versiones que la siguieron y que todavía no han pasado las pruebas.

****Cada branch debe trabajar en tareas específicas, para evitar que estas se repitan en las distintas líneas, logrando que el branch vuelva a integrarse al master en poco tiempo, haciendo que la integración continua sea menos costosa.

**Políticas de fusión de archivos y de etiquetado de acuerdo al progreso de la calidad de los etiquetables.**

En determinados momentos puede ser útil fusionar los branch creados con la línea principal o entre sí.

Si los cambios realizados involucran partes completamente diferentes del código, las versiones se combinan automáticamente.

Pero hay momentos que existen traslapes entre los cambios realizados que además interfieren entre sí. En estos casos el administrador realizara la fusión haciendo un rebase del código junto con los desarrolladores. Así los errores se identificaran rápidamente y los desarrolladores podrán trabajar en resolverlos.

Para estos casos se tendrá en cuenta la etiqueta de cierre (versión a la que se quiere fusionar) y la etiqueta inicial (versión donde nace la rama).

Cada uno de los contadores de etiqueta de versión se modificarán según los criterios definidos en “normas de etiquetado y nombramiento de los archivos”.

**Forma de entrega de los releases**

Luego de realizar la verificación que el software cumple con los requerimientos, y las pruebas hayan sido satisfactorias, podemos decir que estamos en condiciones de lanzar al mercado un release. El cual será un archivo compilado que tendrá la extensión apk.

Para la utilización del mismo, es necesario que el usuario traspase el archivo con extensión .apk a un dispositivo móvil que cuente con el sistema operativo de Android, el cual deberá ser mayor a la versión 2.3.

El software puede ser distribuido mediante la tienda de aplicaciones Google Store, o descargado de internet.

**Herramienta de seguimiento de defectos usada para reportar los defectos descubierto y su estado.**

**Forma de acceso y dirección.**

Github nos proporciona la herramienta Issues para hacer seguimiento de errores, y será la que utilizaremos. Una vez creado el repositorio se tendrá acceso a ella.

La dirección de la herramienta se puede encontrar en la parte superior del documento.

**Historial de versiones**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Nombre | Version | Fecha | Observaciones | Autor |
| Administración-de-las-configuraciones | 1.0.0 | 08/05/2017 | Documento de administración de las configuraciones completo(Entregado) | Gon, Alexander |
| Requerimientos-funcionales-del-sistema | 1.0.0 | 08/05/2017 | Documento de requerimientos del sistema(Entregado) | Orecchini, Stefano |
| Requerimientos-funcionales-del-sistema | 1.1.0 | 21/06/2017 | Actualizo:  -Requerimientos-Pruebas de sistema  -Diagrama de Casos de Uso  -Matrices de trazabilidad | Lenta, Luis |
| Diseño de la arquitectura | 1.0.0 | 22/06/2017 | Documento con el diseño de la arquitectura | Lenta, Luis |
| Diseño e implementación | 1.0.0 | 23/06/2017 | Documento con el diseño y la implementación | Gon, Alexander |
| Requerimientos-funcionales-del-sistema | 1.1.1 | 25/06/2017 | Agrego diseño preliminar de la arquitectura | Lenta, Luis |
| Diseño de la arquitectura | 1.1.0 | 26/06/2017 | Merge con Diseño e implementación | Orecchini, Stefano |