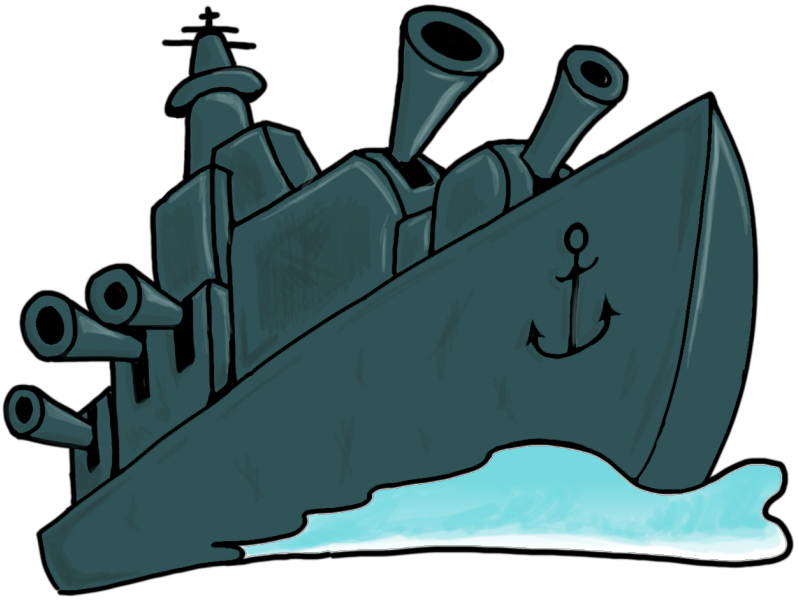


Juego de Batalla Naval

PLAN DE GESTIÓN DE LAS CONFIGURACIONES

Grupo: La Osa que Baila

Integrantes: Martinez, Facundo Jesus

Mugni, Juan Mauricio

Reynoso Choque, Kevin Walter

2022

Historial de Cambios

Versión	Fecha	Resumen	Autor/es
1.0.0	1/5/2022	Primera presentación del informe realizado	Martinez, Facundo Jesus Mugni, Juan Mauricio Reynoso Choque, Kevin Walter

Índice

1 - Introducción

1.1 - Propósito y Alcance

Este documento abarca todo lo que se refiere al Plan de Gestión de las Configuraciones (PGC) para un juego de Batalla Naval. El propósito del PGC es dirigir la estructura de los requerimientos, documentos, software y herramientas usadas para este proyecto.

1.2 - Referencias, acrónimos y glosario

Acrónimo	**Significado**
PGC	Plan de Gestión de Configuraciones
UML	Unified Modeling Language (Lenguaje Unificado de Modelado)
IDE	Integrated Development Environment (Entorno de Desarrollo Integrado)
CRF	Change Request Form (Formulario de Solicitud de Cambios)
CCB	Change Control Board (Junta de Control de Cambios)

1.3 - Herramientas de Administración de Configuraciones

Herramienta	Descripción	Propósito	Link
GitHub	Repositorio remoto para el control de versiones	Repositorio para el proyecto, permite la implementación de ramas	https://github.com/Mauricho/Batalla-Naval
IntelliJ	IDE para codificación en Java	Desarrollo del código utilizado por cada integrante del grupo	
GitHub Actions	Servidor de construcción para integración continua	Realiza la generación manual de compilación y los testeos	https://github.com/Mauricho/Batalla-Naval
GitHub Issues	Herramienta de seguimiento y reporte de defectos	Permite reportar y etiquetar defectos descubiertos y sus estados	https://github.com/Mauricho/Batalla-Naval

1.4 - Miembros del equipo

Los siguientes son los miembros del equipo responsables del desarrollo del proyecto y su correspondiente documentación.

- ☐ Martinez, Facundo Jesus
- ☐ Mugni, Juan Mauricio
- ☐ Reynoso Choque, Kevin Walter

1.5 - Responsabilidades del equipo

A continuación, se detallan las responsabilidades del equipo sobre el proyecto.

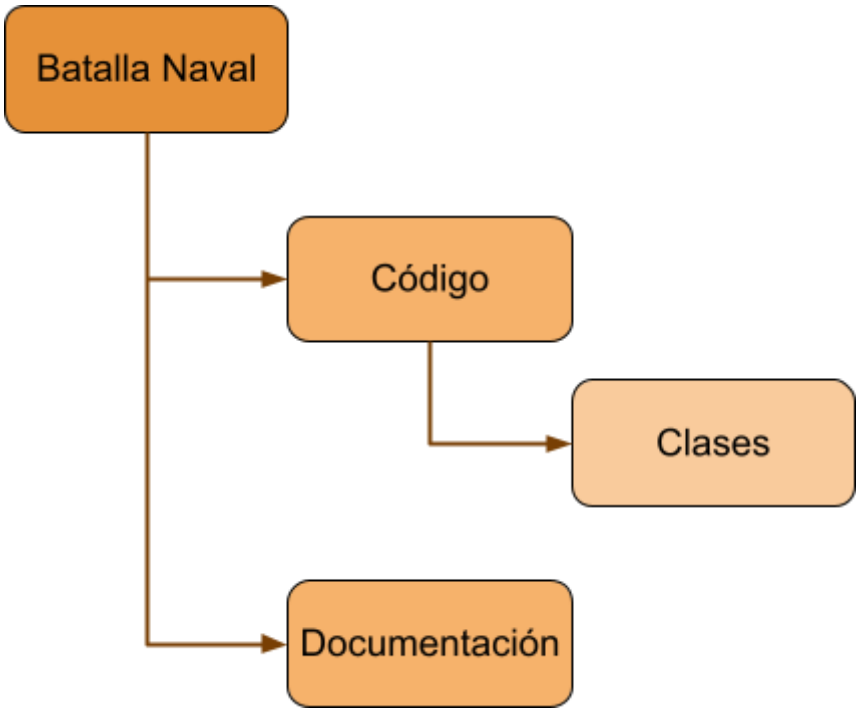
- Definición del proyecto a fin de determinar los recursos
- Definición de las herramientas de administración de configuración
- Configuración correspondiente de las herramientas a implementar
- Codificación por medio del repositorio local y remoto
- Implementación de pruebas automáticas sobre el código desarrollado
- Detección y resolución de defectos o problemas específicos mediante la herramienta especificada para ello
- Establecimiento de la organización del repositorio a fin de garantizar la funcionalidad de cualquier componente a implementar al código
- Cumplimiento de los requerimientos funcionales establecidos para el producto

2 - Esquemas de directorios

El esquema de directorio representa la organización establecida dentro del repositorio remoto del grupo en GitHub. La carpeta principal tiene como nombre “Batalla Naval” en el cual se encuentran todo lo realizado para el trabajo de la materia.

La organización establecida es:

- **Código:** contiene todo lo realizado por el IDE IntelliJ para el proyecto en lenguaje Java.
    - **Clases:** contiene todas las clases definidas para el programa.
- **Documentación:** contiene, por cuestiones de organización, todos los documentos complementarios al proyecto mismo.



3 - Gestión de versiones

3.1 - Normas de etiquetado y de nombramiento de los archivos

Se establece el versionamiento estándar por número (Versionado Semántico) para el etiquetado y nombramiento de versiones.

El formato define un número de versión **X .Y .Z** donde:

- **X** corresponde a la versión mayor
- **Y** corresponde a la versión menor
- **Z** corresponde a la reparación de errores compatibles con versiones anteriores

Se aclaran las siguientes reglas:

1. X, Y y Z **deben** ser números enteros, no negativos y mayores que 0.
2. X puede ser 0 solo para el desarrollo inicial.
3. Una vez que una versión es publicada, los contenidos de esa versión **no deben** ser modificados. Cualquier modificación **debe** ser publicada como otra versión.
4. El número de la versión mayor X es incrementado sólo al haber cumplido con todas las funcionalidades mínimas establecidas para una nueva entrega.
5. El número de la versión menor Y es incrementado al agregar funcionalidades nuevas al programa.
6. El número de la versión de ajuste o de parche Z es incrementado al corregir algún error o comportamiento no deseado del programa.

Para los documentos, se planea implementar el mismo estándar de nombramiento para los archivos adecuado a lo siguiente:

1. El número de la versión mayor X es incrementado sólo al haber completado los requerimientos y correcciones necesarios para la entrega.
2. El número de la versión menor Y es incrementado al agregar información adicional al archivo (ejemplo, gráficos).
3. El número de la versión de ajuste Z es incrementado al corregir secciones del documento existente.

3.2 - Sistema del historial de cambios

Realizar cualquier cambio sobre el código que sea guardado, se enlistara en el repositorio externo (según **1.3**). Sin embargo, se debe llevar el etiquetado correspondiente en base a las normas establecidas a fin de tener disponible palabras claves que describan el nivel de cambio generado (es decir, deben describir el *commit* hecho). Además, la herramienta implementada permite seleccionar, de la lista creada, versiones del código en particular según sea necesario para el desarrollador.

Respecto a la documentación, cualquier cambio conciso que abarque un nivel de las normas establecidas se realizará e implementará sobre el mismo documento y se indicará en el historial de cambios.

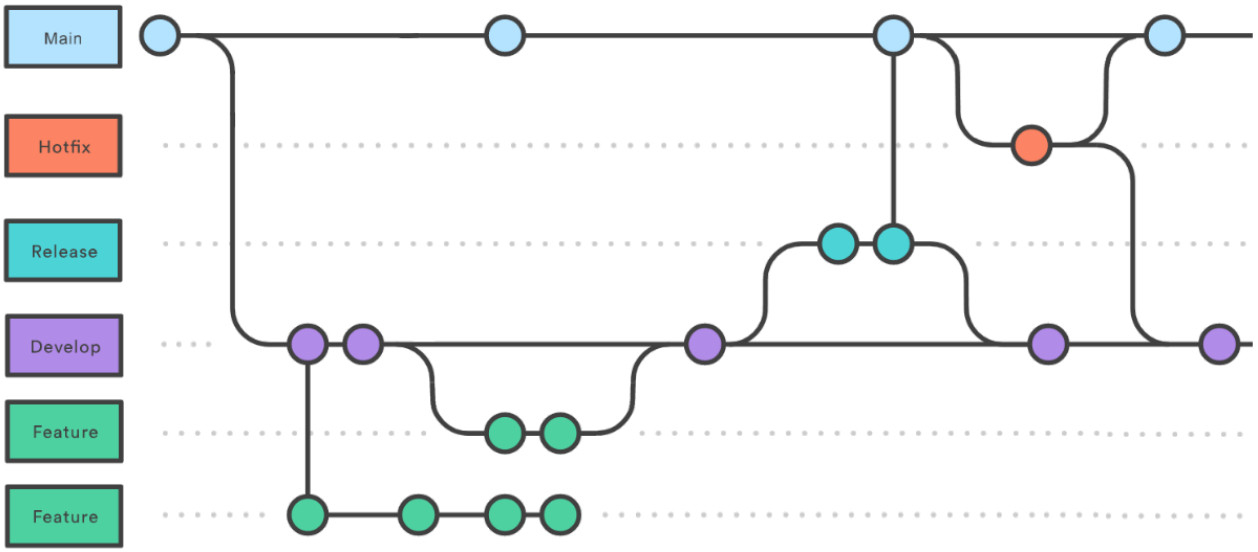
3.3 - Interacción del versionado con el desarrollo

El sistema de gestión de versiones implementado permite que diferentes desarrolladores trabajen simultáneamente sobre el mismo código o componente asegurando que los cambios hechos no interfieran entre ellos.

4 - Gestión de la configuración del código

4.1 - Plan del esquema de ramas

El plan del esquema de ramas se guiará en base al modelo GitFlow como se muestra en el siguiente diagrama.



El esquema consiste en cuatro tipos de ramas las cuales cumplen funciones determinadas.

Main: es la rama principal del proyecto donde contiene las versiones del programa listo para el "release".

Develop: es la rama secundaria del proyecto donde en ella se realiza el *merge* de cualquier funcionalidad a implementar. De ella, salen las ramas o *branches* Feature.

Feature: son ramas de la Develop donde se desarrollan nuevas funcionalidades para el programa.

Release: son ramas creadas a partir del Develop y son un paso previo para la entrega donde se completa y añade la documentación complementaria al programa.

Hotfix: son ramas creadas a partir del Main donde se arreglan bugs detectados. El merge se realiza en el Main y en el Develop de esa misma versión.

4.2 - Política de etiquetado

Durante el desarrollo del trabajo, se trabajará sobre las ramas indicadas en el esquema. A medida que se avance sobre dichas ramas, deben verse identificados como versiones según la política establecida (ver **3.1**) y ser subido al repositorio. Es obligatorio añadir comentarios a fin de indicar los cambios realizados.

4.3 - Política de fusión de ramas

- Primero se trabajará sobre las ramas de Feature hasta haber completado un funcionamiento mínimo sin fallas aparentes.
- Luego, la rama Feature completada se fusionará con la rama Develop.
- Si el programa hasta el momento cumple los requisitos para el *release*, se fusiona la rama Develop con la rama Release.
- En la rama Release, una vez completada toda la documentación, se fusiona con la rama Main teniendo un proyecto listo para la entrega.
- Cualquier error encontrado y reportado, se trabajará en una rama Hotfix (creada a partir de la rama Main de esa versión entregada) y, una vez solucionado, se fusionará con la rama Main y la rama Develop en la misma versión arreglada.

5 - Gestión de entregas

5.1 - Forma de entrega de *releases*

La entrega del producto será por medio de GitHub donde se debe indicar que la rama Main contiene la versión completa para la entrega al cliente.

5.2 - Formato de entrega de *releases*

El formato de entrega elegido es *.zip* el cual debe encontrarse dentro del mismo repositorio y debe estar siempre disponible para descargarse.

5.3 - Instrucciones de instalación

Se debe incluir un archivo *readme.txt* en la carpeta de descarga en GitHub con la información necesaria para la instalación.

6 - Junta de Control de Cambios (CCB)

6.1 - Funciones y objetivos

El CCB es un comité que decide conjuntamente si implementar los cambios propuestos a un proyecto. Estos cambios se plantean según los factores del desarrollo del proyecto como la fase de desarrollo, los tiempos, las metas de calidad, entre otros. Cualquier cambio decidido debe ser comunicado.

La principal función del CCB es agregar, replantear o eliminar funciones, características, requisitos o requerimientos que mejoren el comportamiento del producto (puede ser a nivel gráfico, lógico o de utilidad así como cambios que mejoren la eficiencia del mismo).

Su principal objetivo es mejorar la calidad del producto y asegurar que cumpla las exigencias del cliente en el proyecto.

6.2 - Miembros de la junta

Los miembros del CCB son los indicados en el punto **1.4**.

6.3 - Periodicidad de reuniones

A primera vista, las reuniones (mediante Discord) deben ser al menos una vez cada fin de semana pero se pueden acordar encuentros entre semana según sea la urgencia. Los horarios se establecen entre las tardes/noches (sea cualquier día) según la disponibilidad del equipo completo.

6.4 - Proceso del control de cambios

El proceso de gestión de cambios es iniciado cuando un cliente completa y envía un pedido de cambio describiendo el cambio requerido en el sistema. Esto podría ser un reporte descriptivo y conciso de un error o un pedido para sumar alguna nueva funcionalidad al sistema.

En etapas más avanzadas del proyecto, existe la posibilidad que sea necesario modificaciones en el producto respecto a lo establecido originalmente debido a errores o por tiempos. Cualquier cambio debe ser aprobado por la CCB. Los cambios propuestos pueden ser presentados mediante el CRF.

7 - Gestión de defectos

El seguimiento y detección de errores en el programa desarrollado es mediante la herramienta definida (ver **1.3**) el cual se encuentra implementada dentro del repositorio remoto utilizado para el proyecto. En ella, se pueden realizar comentarios sobre algún tipo de problema. También, al comentario se le puede complementar etiquetas que indican cuál proyecto se hace referencia y/o los tipos de problemas.

Para el desarrollo del proyecto, el grupo lo utilizará para consultas respecto a la codificación y para remarcar posibles defectos. El objetivo es mantener una ayuda mutua continua entre el equipo y resolver problemas cooperativamente.