Tennis Tournament

Configuration Management Plan

(CM_Plan)

Autor: Siampichetti Gino

Version del documento: v1.0.0

Fecha: 17/11/2022

Historial de revisión

Versión	Fecha	Resumen de Cambios	Autor
1.0	19/11/2022	Planificación	Siampichetti Gino
2.0	21/11/2022	Release	Siampichetti Gini

Sumario

1. Introducción

1.1 Propósito y alcance del plan

Este documento abarca el Plan de Manejo de Configuración de *Tennis_Tournament*. El objetivo del mismo es controlar los requisitos de configuración, documentos, software y herramientas usadas en el proyecto, así como registrar y procesar los cambios propuestos al sistema.

1.2 Propósito del plan de gestión de configuraciones

- Asegurar consistencia durante el desarrollo de proyectos de software.
- Mantener la integridad del producto durante todo su ciclo de vida.
- Informar a quien lo requiera sobre el estado de desarrollo del proyecto.
- Crear un historial verificable sobre los estados de trabajo pasados y presente.
- Mejora de los procesos necesarios para llevar a cabo el producto

1.3 Herramientas de control de configuraciones

Herramienta	Propósito	Control de herramienta	Forma de acceso
Github	Servidor para integración continua y herramienta de control de cambios.	Solicitudes de cambio.	https://github.com/ginos1998/tennis_tournament
Github Issues	Control de cambios	Herramienta disponible en Github para el control y administración de todos los requerimientos de cambios, mejoras, y reportes de bugs.	https://github.com/ginos1998/tennis_tournament/issues
StarUML	Diseño de diagramas UML.	La herramienta genera diagramas de tipo UML para el diseño del proyecto.	StarUML https://staruml.io/ UML Plant (IntelliJ)
JUnit	Automatización de pruebas unitarias		IntelliJ (IDE)
SceneBuilder	Agiliza la construcción de la interfaz gráfica	Nos permite crear interfaces gráficas más personalizadas y de forma más intuitiva.	IntelliJ (IDE)
CheckStyle	Permite desarrollar código de mayor calidad	Herramienta para escribir código en Java de acuerdo a los estándares propuestos por Google.	IntelliJ (IDE)

2. ESQUEMA DE DIRECTORIOS

2.1 Scope

La gestión de cambios es un proceso que tiene lugar después de que se identifique y apruebe la documentación, el código fuente o la línea base de hardware del producto. Los cambios incluyen modificaciones internas del planteamiento original documentado debido a los resultados de las simulaciones o las pruebas o a las solicitudes externas de cambios en las características o funciones.

2.2 Estructura de directorios y propósitos

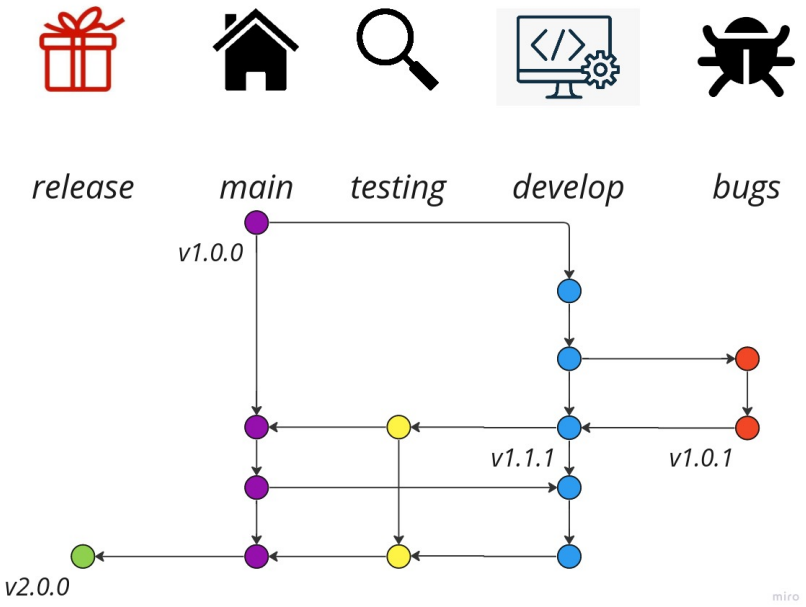
Nombre	Propósito	Link
src	Contiene el código fuente del juego y los recursos como imágenes y audio.	https://github.com/ginos1998/tennis_tournament/tree/main/Tennis_Tournament/src/main
Release	Contiene el archivo ejecutable del juego.	
Documentación del proyecto	Contiene todos los documentos y recursos que se han utilizado para realizar el desarrollo del proyecto.	
Documentación del usuario	Contiene toda la información que pueda llegar a necesitar el usuario.	https://github.com/ginos1998/tennis_tournament/blob/main/README.md

2.3 Normas de etiquetados y nombramiento de archivos

Significado	Nivel	Regla	Ejemplo
Extensión de la primer release	Nuevo Producto	Empieza en **1.0.0**	**1.0.0**
Corrección de bugs	Release de Parche	Incrementa el tercer dígito	1.0.**1**
Nuevas modalidades mínimas	Release menor	Incrementa el segundo dígito y el último dígito se pone en 0.	1.**1**.0
Grandes cambios y funcionalidades grandes o nuevas	Release mayor	Incrementa el primer dígito y los demás toman valor cero	**2**.0.0

3. GESTIÓN DE LA CONFIGURACIÓN DEL SOFTWARE

3.1 Esquema de ramas



Los tipos de ramas que se utilizarán se definen de la siguiente manera:

- Rama **main**: rama de integración principal donde se guardarán todas las funciones publicadas, luego de ser debidamente testeadas.
- Rama **develop**: rama donde se lleva a cabo el desarrollo de nuevas características y/o funcionalidades.
- Rama **test**: Es aquella rama donde se prueba que las funcionalidades básicas para que el programa se ejecute.
- Rama de **bugs**: Es aquella rama donde se solucionan problemas de compilación y test.
- Rama **release**: Representa el código más actualizado que se envía a producción. Cuando el código fuente en main es estable y se ha implementado, todos los cambios se integrarán a release y se etiquetan con el número de versión correspondiente.

4 Gestión de Entregas

4.1 Formato de entrega de releases

Se entregará el release en una carpeta dentro del repositorio de nombre Releases, correspondiente a la versión que se haya seleccionado del proyecto.

4.2 Formato de entrega del instalador

Se va a entregar un ejecutable .jar, para que sea compatible tanto para Windows o Linux.

4.3 Instrucciones mínimas de instalación

Java 19 o superior.