Universidad de Sevilla  
 Escuela Técnica Superior de Ingeniería Informática

**Configuration Management Methodology**

Logotipo

Descripción generada automáticamente

Grado en Ingeniería Informática – Ingeniería del Software  
 Proceso Software y Gestión II

Curso 2022 – 2023

| **Fecha** | **Repositorio** |
| --- | --- |
| 11/2/2023 | https://github.com/gii-is-psg2/psg2-2223-g5-52 |

| **Grupo de prácticas: G5-52** | |
| --- | --- |
| **Autores por orden alfabético** | **Emails** |
| Daniel Arriaza Arriaza | daniel.arriaza.arriaza@gmail.com |
| Guillermo Galeano De Paz | guillermo.galeanodepaz@gmail.com |
| Juan Luis Ruano Muriedas | juanluis.ruano.muriedas@gmail.com |
| Pedro Jesús Ruiz Aguilar | cocode2002@gmail.com |
| José Manuel Ruiz Pérez |  |
| Jesús Solís Ortega | jesussolisortega@gmail.com |

**Índice**

[1. Control de versiones](#_heading=h.gjdgxs) 3

[2. Introducción](#_heading=h.30j0zll) 4

[3. Contenido](#_heading=h.1fob9te) 5

[a) Coding standards](#_heading=h.h4jop6tqo8uc) 5

[b) Políticas de Mensajes de Commit](#_heading=h.wq3lh6lyg45o) 6

[c)](#_heading=h.588fzty4cp1s) 7

[d)](#_heading=h.cu6lgarorkwa) 7

[e)](#_heading=h.i044itzg2xj6) 7

[4. Conclusiones](#_heading=h.3znysh7) 8

[5. Bibliografía](#_heading=h.2et92p0) **9**

## 1. Control de versiones

| Número de revisión | Fecha | Descripción |
| --- | --- | --- |
|  |  |  |

## 2. Introducción

La Gestión de Configuración es un proceso fundamental en el desarrollo de software que permite rastrear y controlar los cambios realizados en el código fuente y otros artefactos durante todo el ciclo de vida del proyecto. En este documento, se describe la metodología de Gestión de Configuración seguida por nuestro grupo, que puede ser aplicada a diferentes proyectos.

## 3. Contenido

### a) Coding standards

Los coding standards son una serie de reglas a seguir por el equipo de trabajo a la hora de crear código. Estas reglas se componen de buenas prácticas a seguir para facilitar la comprensión del código escrito y para mantener la uniformidad del código escrito por diferentes desarrolladores.

Los definidos por el grupo de trabajo G5-52 son los siguientes:

-Uso de camel case:

A la hora de escribir nombres de métodos, clases, interfaces y variables se usará camel case al ser el método al que están habituados los miembros del grupo además de ser la sintaxis usada por java.

-Llaves de sentencias:

Al definir sentencias y métodos se abrirán las llaves en la misma línea en la que se defina, además al cerrar se escribirán en una línea separada exceptuando los else/ else if en los que se cerrará la sentencia del if en la misma línea que en la que se cree.

Ejemplo:

if(...){

…

}else{

…

}

-Nombramiento de variables y métodos:

Durante la creación de nuevas variables o métodos, el nombramiento de estos deberá ser autoexplicativo, es decir, el nombre debe ser tal que al leerlo de suficiente información para comprender su propósito o significado.

-Idioma:

Todo el código deberá ser desarrollado en inglés ya que en la industria del desarrollo software, el inglés es el idioma vehicular. Por tanto, todo comentario o nombre en el código deberá estar en inglés.

-Comentarios:

Evitar el uso de comentarios para mantener el código limpio. Estos se emplearán para aportar la información que se considere necesaria para el entendimiento del funcionamiento de métodos de una complejidad alta. Además al nombrar variables dentro de un comentario se usarán mayúsculas para facilitar la identificación de estas.

### b) Políticas de Mensajes de Commit

Un commit es una instantánea del código fuente y otros artefactos que se realiza para guardar los cambios realizados en el repositorio. Es importante que los mensajes de commit sean claros y concisos para que cualquier persona que lea el historial de cambios pueda entender lo que se hizo en ese commit. A continuación, se describen las políticas de mensajes de commit que seguimos en nuestro grupo:

Estructura de los commits:

* La referencia en el Product Backlog de la tarea: esto permite un mejor seguimiento de las tareas realizadas en función de los requerimientos del cliente.
* Usar "Fixes #x", "Refs #x" para autocerrar, referenciar: Si un commit soluciona un problema o referencia una tarea específica, se debe utilizar la palabra "fixes" o "refs" seguida del número de la tarea correspondiente. Esto permite que la tarea se cierre automáticamente cuando se fusiona el commit en la rama principal.
* Una descripción concisa de lo que se ha hecho. Esto nos permite asegurarnos de que estamos realizando los cambios correctos y que estamos alineados con los objetivos del proyecto.

Esto nos da la siguiente estructura del commit, que a modo de ejemplo quedaría así:

A2.7.b Fixes #10 Commit message policies

Reglas de redacción:

* Intentar que sean títulos breves, en caso necesario añadir descripción: Los títulos de los mensajes de commit deben ser breves y concisos para que sean fáciles de leer y entender. En caso de ser necesario, se puede añadir una descripción detallada de los cambios realizados en el cuerpo del commit.
* Commits en inglés: Los mensajes de commit deben estar escritos en inglés para que sean accesibles a cualquier miembro del equipo o persona interesada en el proyecto.

### c)

### d)

### **e)**

## 4. Conclusiones

En este documento, hemos descrito la metodología de Gestión de Configuración seguida por nuestro grupo. Hemos discutido los estándares de codificación, las políticas de mensajes de commit, la estructura de repositorios y ramas predeterminadas, la estrategia de branching basada en Git Flow y la política de versionado. Al seguir estas prácticas, esperamos ser capaces de controlar los cambios realizados en el código fuente y otros artefactos, lo que nos permitirá desarrollar software de alta calidad de manera eficiente y efectiva. Esperamos que esta metodología pueda ser aplicada en diferentes proyectos para mejorar su Gestión de Configuración y obtener mejores resultados.

# 5. Bibliografía