Lectura - Best Practices

Sitio:	Moodle Academi	Impreso por:	Gadiel Ventura
Curso:	Git Version Control Systems	Fecha:	martes, 16 de enero de 2024, 17:15
Libro:	Lectura - Best Practices		

Descripción

Anterior Siguiente

Tabla de Contenidos

1. Estandar para mensajes de commit

- La estandarización de los mensajes en los commits es sumamente importante ya que es la mejor manera de comunicar el contexto del cambio realizado al resto de los desarrolladores que trabajan en el proyecto.

Estructura convencional para un mensaje de commit

```
<type>: <description>
```

```
[optional body]

[optional footer(s)]
```

Las 7 reglas de un gran mensaje de commit

1. Separa el asunto del cuerpo con una línea en blanco
2. Limita la línea de asunto a 50 caracteres
3. Capitaliza la línea de asunto
4. No termines el asunto con un punto
5. Utiliza el modo imperativo en el asunto

   Una línea de asunto propiamente formada de git commit siempre debe poder completar la siguiente oración:
   Si se aplica, este commit ***<aquí va la línea de asunto>***

   Por ejemplo:

   - *Si se aplica, este commit elimina los métodos obsoletos*
   - *Si se aplica, este commit lanza la versión 2.0*

   Nota: El uso del imperativo es importante solo en la línea del asunto, puedes omitir esta regla al momento de redactar el cuerpo del mensaje.
6. Envuelve el cuerpo del mensaje en 72 caracteres
7. Utiliza el cuerpo para explicar qué y por qué vs cómo

   En la mayoría de los casos, se pueden omitir los detalles sobre cómo se realizó un cambio. Al momento de escribir el cuerpo del mensaje solo se debe de concentrar en aclarar las razones principales del por qué hizo el cambio, la forma en la que las cosas funcionaban antes del cambio (y qué estaba mal en eso), finalizando con la forma en la que las cosas funcionan ahora y por qué decidió resolverlo de la forma en la que la hizo.

Tabla 1. Tipos de cambios aceptados.

Tipo	Descripción
feat	Nueva característica

Tipo	Descripción
fix	Corrección de bugs/errores.
Style	Aplicación de formato, puntos y comas faltantes, etc; sin cambio de código productivo
refactor	Refactorización de código.
perf	Mejoras de rendimiento
test	Agregar o refactorizar pruebas; sin cambio de código productivo
docs	Cambios en la documentación
chore	Mantenimiento regular del código
version	Aumento de versión/ nuevo release; sin cambio de código productivo
ops	Componentes operativos como: deployment, infrastructure
defaults	Cambiar las opciones predeterminadas

2. Branching model

Naming Convention

1. Inicia el nombre con un nombre de grupo. ***bug/… , feature/…., fix/….***
2. Usa un guión o una barra como separadores ***feature/login feature_logout***
3. Rama de Git con nombre de autor

   <author>_<branch-type>_<branch-name>
4. Evita usar sólo números como nombres.
5. Evita nombres descriptivos demasiado largos.