Convención para los Commits en un Repositorio

OBJETIVO: Establecer la forma en que se deben detallar los commits al trabajar en un repositorio.

AUTOR: Oscar Ancán

VERSIÓN: 01102023

La Convenciones para los Commits o Conventional Commits[1] es una especificación para estructurar los mensajes de commit de una manera estandarizada y fácilmente comprensible. Estos mensajes siguen un formato específico que incluye un prefijo que indica el tipo de cambio realizado, seguido de una descripción concisa del cambio. Además, los mensajes de commit pueden contener una línea de cuerpo opcional para proporcionar más detalles y una línea de pie de página opcional para vincular el commit a un problema o solicitud de extracción.

A continuación, se describe el formato general de un mensaje de commit utilizando Conventional Commits, así como una tabla que enumera todos los **prefijos** explicados y otra tabla que muestra todas las **áreas** explicadas, además de un par de ejemplos.

1. Formato General de un Mensaje de Commit con Conventional Commits

El formato general de un mensaje de commit con Conventional Commits consta de tres partes opcionales:

```
<prefijo>[(área opcional)]: <descripción>

<espacio en blanco>

<opcional línea de cuerpo>

<opcional línea de pie de página>
```

Prefijo: Indica la naturaleza del cambio realizado en el commit.

Área: Proporciona contexto adicional sobre el cambio.

Descripción: Breve resumen del cambio.

2. Tabla de Prefijos

Prefijo	Descripción
feat	Para nuevas características (feature).

1 https://www.conventionalcommits.org/es/

Prefijo	Descripción
fix	Para correcciones de errores (bug fixes).
docs	Para cambios en la documentación.
style	Para cambios que no afectan al código (formatting, typos, etc.).
refactor	Para cambios en el código que no corrigen errores ni añaden características.
perf	Para mejoras de rendimiento.
test	Para la adición o modificación de pruebas.
build	Para cambios en el sistema de construcción o dependencias.
ci	Para cambios en la configuración o scripts de integración continua.
chore	Para tareas de mantenimiento que no están relacionadas con cambios en el código fuente.
revert	Para revertir un commit anterior.
deps	Para actualizaciones o cambios en las dependencias del proyecto.
security	Para cambios relacionados con la seguridad del sistema.
init	Para el primer commit de un proyecto o la creación inicial de un módulo.
merge	Para commits de fusiones de ramas.
release	Para commits relacionados con la creación o gestión de versiones del software.
hotfix	Para correcciones rápidas aplicadas a una versión en producción.

3. Tabla de Áreas

Área	Descripción
api	Cambios relacionados con la API del sistema.
ui	Cambios relacionados con la interfaz de usuario.
docs	Cambios en la documentación del proyecto.
config	Cambios en la configuración del sistema.
tests	Cambios en las pruebas o tests del proyecto.
dependencies	Cambios en las dependencias del proyecto.
security	Cambios relacionados con la seguridad del sistema.
performance	Cambios relacionados con mejoras de rendimiento.
database	Cambios relacionados con la base de datos.
frontend	Cambios específicos del frontend de la aplicación.
backend	Cambios específicos del backend de la aplicación.
logging	Cambios relacionados con el registro de eventos.
authorization	Cambios relacionados con la autorización de usuarios.
internationalization	Cambios relacionados con la internacionalización del proyecto.

Área	Descripción
accessibility	Cambios relacionados con la accesibilidad de la aplicación.

4. Ejemplos:

Ejemplo #1

```
feat(controller): Agregar parámetros adicionales a método 'update' en UserController

- Se agregaron parámetros de edad y correo electrónico al método 'update' en el controlador
UserController.
- Se actualizaron las pruebas correspondientes para reflejar los cambios.
```

En este ejemplo:

- feat(controller): Indica que se trata de una nueva característica relacionada con un controlador.
- Agregar parámetros adicionales a método 'update' en UserController: Es una descripción clara y concisa del cambio realizado.
- La sección del cuerpo proporciona más detalles sobre los cambios específicos realizados.
- No se incluye una línea de pie de página en este caso, ya que no hay ninguna referencia a un problema específico o un número de solicitud de extracción relacionado.

Ejemplo #2

```
feat(ui): Actualizar servicio de conexión a API en UserService

- Se actualizó el método 'getUserData' en el servicio UserService para incluir la nueva
funcionalidad de conectarse a la API utilizando la biblioteca Axios.
- Se implementó manejo de errores para casos de fallo en la conexión con la API.
- Se actualizó la documentación del servicio para reflejar los cambios realizados.
```

En este ejemplo:

- feat(ui): Indica que se trata de una nueva característica relacionada con la interfaz de usuario.
- Actualizar servicio de conexión a API en UserService: Es una descripción clara del cambio realizado en el servicio UserService.
- La sección del cuerpo proporciona detalles específicos sobre los cambios realizados, incluyendo la actualización del método, la implementación del manejo de errores y la documentación.