# Conventional Commits & Gitmoji

Florencia Yáñez G.

# 1 ¿Qué son?

Son convenciones o estándares utilizados para escribir mensajes de commit en sistemas de control de versiones como Git.

# 2 Conventional Commits

# // ¿Para qué sirven?

### LEGIBILIDAD

Imponen una estructura clara y predecible para los mensajes de commit.

### COLABORACIÓN

Facilita a otros miembros del equipo (e incluso a ti mismo en el futuro) entender el propósito y alcance de cada commit.

### AUTOMATIZACIÓN

Dado que siguen un formato específico, es más fácil automatizar la generación de changelogs e integrarse con versionado semántico.

# Estructura

- **Commit Summary:**
  - **Tipo**: tipo de cambio que se realiza en el commit. Obligatorio.
  - **Scope**: scope en el que se lleva a cabo el commit.
  - **!**: especifica que el commit rompe partes del código.
  - **Descripción**: breve descripción de los cambios. Obligatorio.

- **Commit Description**:
  - **Cuerpo**: provee más información sobre los cambios. Su formato es libre y puede consistir de varios párrafos.
  - **Footers**: referencia IDs de issues, números de pull request, notas de breaking changes, etc. Opcional excepto en el caso de breaking changes.

```
tipo(scope)!: descripción
Cuerpo
Footer: string
```

# Tipos de Conventional Commits

- **feat**: introduce una nueva feature.
- **fix**: arregla un bug.
- **docs**: solo cambia documentación.
- **style**: cambios que no afectan el significado del código.
- **refactor**: cambio que no arregla un bug ni agrega una feature.
- **perf**: cambio que mejora la performance.
- **test**: agregar tests faltantes o arreglar tests existentes.
- **chore**: cambios al proceso de build o herramientas auxiliares y librerías.
- **build**: cambios que afectan el sistema de build o dependencias externas.
- **revert**: revertir un commit anterior.

Además de los tipos listados, un equipo o proyecto puede introducir tipos adicionales que sean específicos para su workflow o requerimientos.

La clave principal es mantener una historia de commits consistente y descriptiva, así que los tipos usados pueden ser adaptados a las necesidades del proyecto.

# Ejemplo

```
fix(service)!: change http conference errors

Change http conference errors to be more specific.

BREAKING CHANGE: tests for conference failing
```

# 3 Gitmoji

# // ¿Para qué sirven?

### IDENTIFICACIÓN RÁPIDA

Añaden un elemento visual, lo que puede hacer más fácil y rápida la identificación del propósito o naturaleza del commit.

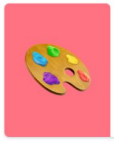
### ÁNIMO

Puede hacer que la revisión de commits sea más agradable y menos monótona, lo que puede mejorar el ánimo y la cultura del equipo.


### ENTORNOS MULTILINGÜES

Pueden servir como un lenguaje universal para ciertos conceptos. Reduce barreras lingüísticas y asegura que todos entiendan el propósito general de un commit.

# Ejemplos



**:art:**

Improve structure / format of the code.

**:zap:**

Improve performance.

**:fire:**

Remove code or files.

**:bug:**

Fix a bug.

**:ambulance:**

Critical hotfix.

**:sparkles:**

Introduce new features.

**:memo:**

Add or update documentation.

**:rocket:**

Deploy stuff.

**:lipstick:**

Add or update the UI and style files.

**:tada:**

Begin a project.

**:white_check_mark:**

Add, update, or pass tests.

**:lock:**

Fix security or privacy issues.

**:closed_lock_with_key:**

Add or update secrets.

**:bookmark:**

Release / Version tags.

**:rotating_light:**

Fix compiler / linter warnings.

Se puede usar más de un emoji por commit

# 4

# ¿Conventional Commits o Gitmoji?

# Comparación

## Conventional Commits

- Estructura un poco compleja y rígida.
- Enfocado en estandarización.
- Adoptado en proyectos que buscan alto grado de automatización y estandarización en sus procesos de desarrollo.

## Gitmoji 😜

- Estructura simple y maleable.
- Enfocado en mejorar visibilidad.
- Popular en proyectos y equipos que valoran la expresividad visual y comunicación informal.

¿Y podemos usar los DOS?

¡¡¡Si, se puede!!!

# Conventional Commits + Gitmoji

Esta combinación puede proporcionar tanto la claridad y estructura de Conventional Commits como el atractivo visual y la rápida identificación de cambios que ofrece Gitmoji.

**Ejemplo**:

```
✨ feat(scope): add 'remember me' feature
```

**Consideraciones:**

**Claridad**: asegurar que la adición de emojis no comprometa la claridad del mensaje de commit, el objetivo es mejorar la comunicación, no hacerla más confusa.

**Compatibilidad**: tener en cuenta la compatibilidad con herramientas de línea de comandos y entornos donde los emojis pueden no renderizarse correctamente.

**Simplicidad**: aunque la combinación ofrece ventajas, también puede añadir cierta complejidad, hay que considerar si los beneficios superan los posibles inconvenientes para el equipo y proyecto.

# Evidencia 🕵🏻

| Conventional Commits + Gitmoji | Sin estándar |
| --- | --- |
| ✨ feat(auth): Implement OAuth2 flow | Added login stuff |
| 🐛 fix(server): Resolve memory leak issue | fixed something |
| 📝 docs: Update README with new endpoints | README changes |
| 🔥 chore: Remove deprecated methods | Removed old code |
| 🚀 perf(database): Optimize query logic | Made db faster |
| 🧪 test: Add tests for user service | More tests |

# Demo

Puedes describir tus cambios a Github Copilot, GenAI o ChatGPT para que te sugiera un Conventional Commit y/o Gitmoji.

# Referencias


Conventional Commits https://www.conventionalcommits.org/en/v1.0.0/

Gitmoji https://gitmoji.dev/

# Gracias.