Políticas de Nomenclatura para el Equipo de Desarrollo

1. Introducción

Este documento establece las políticas de nomenclatura que buscan garantizar la consistencia, claridad y mantenibilidad en el código desarrollado por el equipo. La adherencia a estas políticas es crucial para fomentar un entorno de desarrollo eficiente y colaborativo.

2. Nomenclatura de Bases de Datos

- **Tablas**: Deben ser plurales, en minúsculas y descriptivas del contenido que almacenan. Ejemplo: `users` para almacenar información sobre usuarios, `products` para almacenar información sobre productos.
- **Campos**: Deben ser descriptivos, en minúsculas y separados por guiones bajos. Además, deben ser descriptivos del tipo de datos que almacenan. Ejemplo: `first_name` para almacenar un nombre de persona.

3. Nomenclatura de Variables

- Deben ser descriptivas y seguir la convención de nombres del lenguaje de programación. Ejemplo (en JavaScript con camelCase): `firstName`.
- Deben ser lo suficientemente cortas para ser fáciles de leer y escribir, pero lo suficientemente largas para ser descriptivas.

4. Nomenclatura de Funciones

- Los nombres de las funciones deben ser verbos que describan la acción que realizan. Ejemplo: `calculateSum`, `getUserName`.
- Deben ser lo suficientemente cortos para ser fáciles de leer y escribir, pero lo suficientemente largos para ser descriptivas.

5. Nomenclatura de Clases

- Los nombres de las clases deben ser sustantivos y comenzar con una letra mayúscula. Ejemplo: `Product`, `User`.
- Deben ser descriptivas del propósito de la clase. Ejemplo: `Product` para representar un producto, `User` para representar un usuario.

6. Nomenclatura en Git

- **Branches**: Deben ser descriptivas y reflejar el propósito del branch. Ejemplo: `feature/user-authentication`, `bugfix/payment-processing`.
- **Commits**: Deben ser claros y concisos, describiendo los cambios realizados. Ejemplo: “Add user authentication”, “Fix payment processing bug”.
- **Mensajes de Commit**: Deben seguir una estructura clara y concisa para proporcionar información sobre los cambios realizados (Conventional Commits). Ejemplo:

```
feat: Add user authentication
Adds the ability for users to authenticate with the application.
```

- **Etiquetas (Tags)**: Cuando sea necesario marcar versiones o hitos importantes, se deben utilizar etiquetas descriptivas. Ejemplo: `git tag v1.0 -m "Versión inicial estable"`.

7. Conclusiones

La adhesión a estas políticas de nomenclatura es esencial para lograr un código claro, coherente y fácilmente mantenible. Al seguir estas directrices, el equipo mejora la legibilidad del código, facilita la colaboración y contribuye a la reducción de errores.

8. Complementaciones

Además de las políticas mencionadas anteriormente, se pueden agregar las siguientes recomendaciones para mejorar la consistencia y claridad del código:

- **Usar un estándar de codificación**: Un estándar de codificación establece reglas para la sintaxis, el estilo y la estructura del código. Al utilizar un estándar, el equipo puede garantizar que el código sea consistente y fácil de entender.

- **Usar comentarios**: Los comentarios son una forma de proporcionar información adicional sobre el código. Pueden utilizarse para explicar el propósito del código, describir la lógica de funcionamiento o proporcionar información sobre el estado del código.
- **Documentar el código**: La documentación proporciona información sobre el código, como su propósito, uso y funcionamiento. Al documentar el código, el equipo puede ayudar a los desarrolladores a comprender y utilizar el código de manera más eficaz.