Políticas de Nomenclatura para el Equipo de Desarrollo

Introducción

Este documento establece las políticas de nomenclatura que deben seguirse en la compañía para mantener un código consistente, legible y fácil de mantener. La consistencia en la nomenclatura es fundamental para asegurar la calidad del código y facilitar la colaboración entre los miembros del equipo.

1. Nomenclatura de Bases de Datos

1.1. Tablas

- Las tablas deben nombrarse en **plural**.
- Usar **snake_case** (letras minúsculas y guiones bajos).
- Ser descriptivo y conciso.
- Los nombres deben ser en inglés.

Ejemplo:

- Correcto: `clients`, `full_name`
- Incorrecto: `client`, `FullName`

1.2. Columnas

- Las columnas deben nombrarse en **snake_case**.
- Deben describir claramente los datos que contienen.
- Deben escribirse en **singular**.

Ejemplo:

- Correcto: `full_name`, `contact`
- Incorrecto: `FullName`, `contacts`

1.3. Índices

- Deben seguir el formato `idx_<tabla>_<columna(s)>`.

Ejemplo:

- Correcto: `idx_clients_document_id`
- Incorrecto: `index_clients_documente_id`

1.4. Claves Primarias y Foráneas

- Las claves primarias deben ser `id`.

  Ejemplo:

  - Correcto: `id`,
  - Incorrecto: `client_Id`

- Las claves foráneas deben seguir el formato `<tabla_origen>_id`.

  Ejemplo:

  - Correcto: `clients_id`
  - Incorrecto: `ClientsProductsId`

2. Nomenclatura de Código

2.1. Variables

- Usar **camelCase** para variables locales y parámetros.
- Ser descriptivo pero conciso.

Ejemplo:

- Correcto: `firstName`, `orderTotal`
- Incorrecto: `firstname`, `ordertotal`

2.2. Funciones

- Usar **camelCase**.
- Los nombres deben ser verbos o frases verbales que describen lo que hace la función.

Ejemplo:

- Correcto: `getUserById`, `calculateTotal`
- Incorrecto: `getuserbyid`, `calculate_total`, `GetUserById`

2.3. Clases

- Usar **PascalCase**.
- Los nombres deben ser sustantivos que describen claramente la responsabilidad de la clase.

Ejemplo:

- Correcto: `UserController`, `OrderService`
- Incorrecto: `usercontroller`, `order_service`, `Orderservice`

2.4. Archivos y Directorios

- Los nombres de archivos deben usar **kebab-case**.
- Los nombres de directorios, de ser posible deben ser de 1 sola palabra, sino, deben ser en **kebab-case**.

Ejemplo:

- Correcto: `user-controller.ts`, `order-service.ts`, `models/`, `controllers/`
- Incorrecto: `userController.ts`, `orderService.ts`, `Models/`, `Controllers/`

3. Nomenclatura en Git

3.1. Ramas

- Usar **kebab-case**.
- El nombre de la rama debe ser en inglés.
- Deben iniciar con feature/bugfix/hotfix dependiendo de su propósito.
- Las ramas deben ser descriptivas y reflejar el propósito o la tarea.

Ejemplo:

- Correcto: `feature/add-user-authentication`, `bugfix/fix-order-total-calculation`
- Incorrecto: `FeatureAddUserAuthentication`, `BugFixFixOrderTotalCalculation`

3.2. Commits

- Los mensajes de commit deben ser en inglés.
- La primera línea debe ser una descripción concisa en **imperativo.**
- Incluir una descripción más detallada en las siguientes líneas si es necesario.

Ejemplo:

Correcto:

```
Add user authentication

Implement OAuth2 authentication for users, including login and
registration flows.
```

Incorrecto:

```
Added user authentication

Implemented OAuth2
```

4. Documentación

4.1. Documentación

- Usar herramientas como para documentar funciones, clases y módulos, por ejemplo JSDoc para documentar código escrito en Javascript.
- Asegurarse de que la documentación esté actualizada y sea precisa.
- Contar con una herramienta de trabajo como Confluence para poder documentar el funcionamiento, requisitos, respuesta y errores de los endpoint de cada uno de los servicios.

4.2. Swagger

- Implementar Swagger en los servicios para que los diferentes stakeholders (como testers, product managers, devs, etc.) puedan explorar y entender cómo funciona la API.