

## Documento de Estilos de Código

Este documento establece ciertas normas y convenciones que deben de seguirse para el proyecto “EcoHarmony Park”, el cual es desarrollado con el lenguaje de programación JavaScript junto con CSS y HTML, con React como biblioteca principal.

Indice	hoja
1. Estructura del Proyecto	1
2. Nomenclatura	2
3. Formato y Espaciado	2
4. Comentarios	3
5. Buenas Prácticas con JavaScript	3
6. Buenas Prácticas con HTML	3
7. Buenas Prácticas con CSS	3
8. Buenas Prácticas con React	4

## Estructura del Proyecto

- Mantener una estructura clara y coherente para archivos y carpetas.
- En caso de ser necesario un cambio en la estructura comunicarlo al encargado de SCM.
- Utilizar nombres descriptivos para cada componente y módulo.
- Evitar la redundancia de código, reutilizar si es posible.

Estructura modelo para iniciar:

```
/mi-proyecto
|
├── /frontend
|   ├── /public
|   └── /src
```

```
| | └─ /components
| | └─ /imgs
| | └─ /assets
| | └─ App.js
| | └─ App.css
| | └─ Index.css
| |   └─ main.js
| └─ package.json
|
└─ /backend
  | └─ /node_modules
  | └─ /config
  | └─ /package.json
  |   └─ server.js
└─ .gitignore
└─ Documento de estilo de código
```

## Nomenclatura

- Utilizar camelCase para nombrar variables y funciones.
- Utilizar PascalCase para nombrar clases y componentes.
- Utiliza nombres descriptivos y significativos para variables, funciones y componentes.
- Evitar el uso de abreviaciones en nombres de variables o funciones.

## Formato y espaciado

- Mantener una indentación consistente.
- JavaScript:
  - Usar punto y coma (;) al final de cada declaración.
  - Usar comillas simples (') para cadenas de texto.
- CSS:
  - Mantener una línea en blanco entre reglas de estilo.

- HTML:
  - Asegurarse de cerrar correctamente las etiquetas.

## **Comentarios**

- Utilizar comentarios para explicar el propósito y funcionamiento del código cuando sea necesario aclararlo

## **Buenas prácticas con JavaScript:**

- Usar const por sobre let siempre que se pueda.
  - const para valores constantes.
  - let para variables que pueden cambiar.
  - Evitar el uso de var.
- Uso de arrow functions si es posible para mantener el código más claro y conciso.

## **Buenas prácticas con HTML:**

- Declarar el `<!DOCTYPE html>` para una mejor compatibilidad entre navegadores.
- Cerrar cada etiqueta en el orden correcto.
- Evitar el uso excesivo de etiquetas `<div>`, usando etiquetas como `<header>`, `<main>`.

## **Buenas prácticas con CSS:**

- Uso de Flexbox para organizar los elementos de forma eficiente, para manejar layouts en lugar de utilizar posicionamiento absoluto o flotantes.
- Diseño responsive mediante media queries.

## Buenas prácticas con React

- Los componentes deben ser pequeños, reutilizables y cumplir con una sola responsabilidad. Evitar hacer componentes que manejen múltiples tareas.
- Utilizar componentes funcionales siempre que sea posible en lugar de Clases.
- Usar los React hooks como `useState`, `useEffect`, `useContext`, entre otros.