Documento de Estilo de Código

Proyecto: EcoHarmony Park

Materia: Gestión del Software como Producto

Grupo: ISW - Grupo 3

Lenguajes utilizados: React + TypeScript (Front-end) | Node.js (Back-end)

Objetivo

Definir convenciones y buenas prácticas para mantener un estilo de código coherente, legible y mantenible a lo largo de todo el proyecto, tanto en el front-end como en el back-end.

Organización del Proyecto

Front-end (`/frontend/src/`)

Carpeta	Descripción
assets/	Imágenes y recursos estáticos
components/	Componentes reutilizables (React)
context/	Contexto de la transacción
pages/	Vistas principales de la aplicación
services/	Funciones que interactúan con la API
styles/	Archivos CSS o módulos de estilo
Archivos raíz	Punto de entrada de la app y configuración base

Back-end (`/backend/API/`)

Carpeta	Descripción
config/	Configuraciones generales (ej: entorno, puerto, variables)
Db/	Conexión y configuración de la base de datos
Libs/	Bibliotecas auxiliares o utilitarias

Middlewares/	Validaciones y control de errores
Rutas/	Definición de rutas (endpoints)
Esquemas/	Modelos de datos (schemas)
servicios/	Lógica de negocio, interacción con la DB
index.js	Punto de entrada del servidor

🔤 Convenciones de Nomenclatura

Elemento	Convención	Ejemplo
Variables y funciones	camelCase	obtenerHorario, usuarioId
Clases y componentes	PascalCase	HorarioForm, EventoCard
Archivos	kebab-case o camelCase	horario-form.tsx, authService.ts
Constantes	MAYÚSCULAS_CON_GUIONES	API_URL, TOKEN_KEY

✍️ Estilo de Código

- Indentación: 4 espacios
- Longitud máxima de línea: 100 caracteres
- Comillas: simples ' ' (consistencia entre front y back)
- Punto y coma: opcional
- Espacios: siempre después de comas y operadores
- Saltos de línea: entre funciones y bloques lógicos
- Funciones puras y pequeñas siempre que sea posible

💡 Buenas Prácticas

Front-end (React + TS)

- Separar lógica del render
- Tipar todas las props, estados y funciones
- Usar useEffect y useState de forma controlada
- Evitar lógica dentro del JSX (usar helpers)
- Manejar errores de API en services

Back-end (Node.js)

- Separar rutas, controladores y servicios
- Validar entradas en middleware
- Usar try/catch en controladores

- Escribir funciones reutilizables en Libs
- No dejar console.log() en producción

🔧 Herramientas de Formato

- ESLint con base en Airbnb Style Guide
- Prettier para autoformateo
- Extensiones recomendadas: ESLint, Prettier, TypeScript Hero, VSCode Icons

📚 Referencias

- https://github.com/airbnb/javascript
- https://www.typescriptlang.org/docs/handbook/intro.html
- https://react.dev/learn
- https://github.com/goldbergyoni/nodebestpractices