Documento de Estilos de Código

Este documento establece pautas y convenciones para el desarrollo de código en el proyecto “Tango App”, con el objetivo de mejorar la legibilidad, mantenibilidad y colaboración del código.

1. Convenciones Generales

Utiliza nombres descriptivos y significativos para variables, funciones y componentes.

Mantén el código limpio y organizado, siguiendo la estructura y patrones establecidos en el proyecto.

Usa comentarios para explicar el propósito y funcionamiento del código cuando sea necesario.

2. JavaScript / TypeScript

2.1. Nomenclatura

Utiliza camelCase para nombrar variables y funciones.

Utiliza PascalCase para nombrar clases y componentes.

2.2. Importaciones y rutas de módulos

Usa importaciones absolutas en lugar de rutas relativas siempre que sea posible.

Organiza las importaciones en bloques separados, primero las importaciones de paquetes externos, seguidas de las importaciones internas del proyecto.

Agrupa las importaciones relacionadas entre sí y separa cada grupo con una línea en blanco.

typescript

import React from "react";

import { useRouter } from "next/router";

import Button from "@/components/Button";

import useBudgets from "@/components/hooks/useBudgets";

2.3. Declaración de variables

Utiliza const para declarar variables que no serán reasignadas y let para aquellas que sí lo serán.

Evita la declaración de variables globales cuando sea posible, prefiriendo el ámbito de bloque.

typescript

const username = "John";

let age = 30;

2.4. Estilo de código

Utiliza espacios en lugar de tabulaciones para la indentación.

Usa dos espacios para cada nivel de indentación.

Limita la longitud de las líneas de código a 80-100 caracteres para mejorar la legibilidad.

typescript

const greet = (name: string) => {

console.log(`Hello, ${name}!`);

};

3. React

3.1. Componentes

Utiliza componentes funcionales siempre que sea posible.

Define los prop-types de los componentes para documentar su API.

Agrupa las props relacionadas en interfaces para una mejor organización y documentación del código.

typescript

import React from "react";

interface ButtonProps {

onClick?: () => void;

className: string;

children: React.ReactNode;

}

const Button: React.FC<ButtonProps> = ({ onClick, className, children }) => {

return <button onClick={onClick} className={className}>{children}</button>;

};

export default Button;

3.2. Estilos

Utiliza clases de CSS para estilizar componentes en lugar de estilos en línea siempre que sea posible.

Utiliza classNames o plantillas de cadena para generar nombres de clases de manera dinámica.

javascript

import React from "react";

import classNames from "classnames";

const MyComponent = ({ isActive }) => {

const buttonClasses = classNames("button", {

"active": isActive,

"disabled": !isActive,

});

return <button className={buttonClasses}>Click Me</button>;

};

Conclusiones

Estas son solo algunas de las pautas generales que deben seguirse al escribir código en el proyecto. Se alienta a todos los miembros del equipo a revisar este documento periódicamente y contribuir con sugerencias para mejorar las convenciones de codificación.