Proyecto React con CSS Modular

Este proyecto es un boilerplate para comenzar una aplicación React usando Vite y un enfoque de CSS modular sin dependencias externas como Tailwind. El proyecto está estructurado con un enfoque "mobile first" e incluye soporte para temas claro y oscuro.

Estructura de archivos

```
src/
├── App.jsx                 # Componente principal
├── App.css                 # Estilos del componente principal
├── main.jsx                # Punto de entrada de la aplicación
├── theme.css               # Variables CSS para los temas claro y oscuro
├── ThemeProvider.jsx       # Contexto de React para el manejo del tema
├── ThemeToggle.jsx         # Componente para alternar entre temas
└── ThemeToggle.css         # Estilos para el botón de alternancia
```

Características

- Estructura base con contenido centralizado
- Hero banner con barra de navegación debajo
- Diseño "mobile first" con media queries para dispositivos más grandes
- Sistema de temas claro/oscuro con persistencia en localStorage
- Componentes para alternar el tema
- CSS modular sin dependencias externas
- Variables CSS para personalización fácil

Paleta de colores

Puedes personalizar la paleta de colores modificando las variables CSS en el archivo `theme.css`:

- Para el tema claro, edita las variables en `:root`
- Para el tema oscuro, edita las variables en `[data-theme='dark']`

Instrucciones de uso

1. Clona este repositorio
2. Instala las dependencias con `npm install`
3. Ejecuta el servidor de desarrollo con `npm run dev`
4. Abre http://localhost:5173 en tu navegador

Personalización

Colores

Modifica las variables de color en `theme.css` para adaptar la paleta a tus necesidades.

Fuentes

Cambia las variables `--font-primary` y `--font-heading` en `theme.css`.

Componentes

Los componentes base como tarjetas, botones y secciones están predefinidos en `App.css` y puedes modificarlos según tus necesidades.

Desarrollo con CSS Modular

Este proyecto sigue un enfoque modular para el CSS:

1. Define variables globales en `theme.css`
2. Utiliza clases específicas para cada componente
3. Aplica el enfoque "mobile first" con media queries para pantallas más grandes
4. Mantiene las variables de tema separadas de los estilos de layout

Ampliación del proyecto

Para expandir este proyecto, puedes:

1. Crear componentes adicionales en archivos separados
2. Definir estilos específicos para cada componente
3. Integrar enrutamiento con React Router cuando necesites más páginas
4. Agregar gestión de estado con Context API o Redux cuando tu aplicación crezca

Licencia