SUPER-PROMPT PARA REPLIT — Librería de Componentes React + TypeScript (AI‑friendly)

===================================================================================

Objetivo

--------

Generar una \*\*librería de componentes UI\*\* en React + TypeScript, publicada en GitHub (Packages o `git+ssh`), con documentación mínima pero accionable para humanos y \*\*IA\*\* (archivos cortos “AI‑friendly”). La librería debe:

- Responder al \*\*tema global\*\* (`dark`/`light`) automáticamente.

- Usar i18n con \*\*prioridad local-first\*\* (primero diccionario del componente; si no existe la key, usar global; si no, fallback a la key interpolada).

- Exponer una \*\*estructura clara\*\* (repo, build, tests, stories, docs, ai-manifest) para integración inmediata.

- Incluir \*\*CI/CD\*\* con `semantic-release` (versionado semántico) y publicación.

- Ser \*\*portable\*\* y amigable a tree-shaking (ESM + `d.ts`).

⚠️ Nota sobre la \*\*estructura del componente\*\*:

> \*\*No detallar aquí\*\*. Para la estructura interna (carpetas `types/`, `hook/`, `provider/`, `view/`, `utils/`, i18n local-first, visibilidad, y manejo de tema dark/light) \*\*usaremos el otro archivo\*\* que acabamos de crear:

> \*\*“Guía — Componente Avanzado React + TypeScript (local-first)”\*\* (`Guia\_Componente\_React\_TS\_local-first.docx`).

Metas

-----

- \*\*Contexto IA reducido\*\*: guías, manifiestos y snippets que permitan a la IA implementar sin leer todo el código.

- \*\*Uso inmediato\*\*: Quickstart, checklist de integración, ejemplos mínimos y tabla de props auto-documentadas.

- \*\*Escalable\*\*: versionado semántico, tests, Storybook, tokens de diseño, barrel exports.

- \*\*Repos reproducibles\*\*: scripts estándar, builds ESM, types separados, CI estable.

Placeholders (busca y reemplaza)

--------------------------------

- `{{ORG}}` (e.g., `fremitech`)

- `{{REPO}}` (e.g., `ui-library`)

- `{{SCOPE}}` (e.g., `@fremitech`)

- `{{PKG\_NAME}}` (e.g., `ui`)

- `{{REGISTRY\_URL}}` (e.g., `https://npm.pkg.github.com`)

- `{{NODE\_VERSION}}` (e.g., `20`)

- `{{DEFAULT\_BRANCH}}` (e.g., `main`)

Estructura del repositorio

--------------------------

{{REPO}}/

.github/

workflows/

ci.yml

release.yml

.npmrc

.gitignore

LICENSE

README.md

README\_AI.md

CONTRIBUTING.md

CHANGELOG.md

CODE\_OF\_CONDUCT.md

CODEOWNERS

package.json

tsconfig.json

tsconfig.build.json

vite.config.ts

storybook/

main.ts

preview.ts

src/

index.ts

components/

Button/

Button.tsx

Button.types.ts

Button.stories.tsx

index.ts

\_\_tests\_\_/Button.test.tsx

README\_AI\_Component.md

TextField/

...

hooks/

useDebouncedValue.ts

index.ts

theme/

tokens.ts

ThemeProvider.tsx

index.ts

utils/

clsx.ts

index.ts

docs/

quickstart.md

integration-checklist.md

migration-guide.md

ai/

manifest.json

component-map.json

snippets/

button.install.md

button.usage.md

form.pattern.md

Configuración clave

-------------------

### package.json

- `"type": "module"`

- Builds ESM + types (`tsc` + Vite).

- Scripts:

- `"dev": "storybook dev -p 6006"`

- `"build": "tsc -p tsconfig.build.json && vite build"`

- `"lint": "eslint ."`

- `"test": "vitest run"`

- `"release": "semantic-release"`

- `"prepublishOnly": "npm run build"`

- `publishConfig.registry={{REGISTRY\_URL}}`

- `name="{{SCOPE}}/{{PKG\_NAME}}"`

Ejemplo:

{

"name": "{{SCOPE}}/{{PKG\_NAME}}",

"version": "0.1.0",

"type": "module",

"main": "dist/index.js",

"module": "dist/index.js",

"types": "dist/index.d.ts",

"scripts": {

"dev": "storybook dev -p 6006",

"build": "tsc -p tsconfig.build.json && vite build",

"lint": "eslint .",

"test": "vitest run",

"release": "semantic-release",

"prepublishOnly": "npm run build"

},

"publishConfig": { "registry": "{{REGISTRY\_URL}}" }

}

### .npmrc

@{{SCOPE:sin@}}:registry={{REGISTRY\_URL}}

; En CI exportar NODE\_AUTH\_TOKEN con acceso a GitHub Packages

### Workflows de GitHub Actions

\*\*.github/workflows/ci.yml\*\*

name: CI

on:

push:

branches: [ "{{DEFAULT\_BRANCH}}" ]

pull\_request:

branches: [ "{{DEFAULT\_BRANCH}}" ]

jobs:

build-test-lint:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- uses: actions/setup-node@v4

with:

node-version: "{{NODE\_VERSION}}"

registry-url: "{{REGISTRY\_URL}}"

- run: npm ci

- run: npm run build

- run: npm test

- run: npm run lint

\*\*.github/workflows/release.yml\*\*

name: Release

on:

push:

branches: [ "{{DEFAULT\_BRANCH}}" ]

jobs:

release:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

- uses: actions/setup-node@v4

with:

node-version: "{{NODE\_VERSION}}"

registry-url: "{{REGISTRY\_URL}}"

- run: npm ci

- run: npm run build

- env:

GITHUB\_TOKEN: ${{ secrets.GITHUB\_TOKEN }}

NODE\_AUTH\_TOKEN: ${{ secrets.NODE\_AUTH\_TOKEN }}

run: npx semantic-release

### tsconfig.json (mínimo)

{

"extends": "./tsconfig.build.json",

"compilerOptions": {

"skipLibCheck": true

}

}

### tsconfig.build.json (ESM + d.ts)

{

"compilerOptions": {

"target": "ES2020",

"module": "ESNext",

"moduleResolution": "Bundler",

"declaration": true,

"emitDeclarationOnly": true,

"jsx": "react-jsx",

"outDir": "dist",

"strict": true,

"baseUrl": ".",

"paths": { "\*": ["./types/\*"] }

},

"include": ["src"]

}

### vite.config.ts (build ESM)

import { defineConfig } from 'vite';

import dts from 'vite-plugin-dts';

export default defineConfig({

build: {

lib: {

entry: 'src/index.ts',

formats: ['es'],

fileName: () => 'index.js'

},

rollupOptions: {

external: ['react', 'react-dom']

}

},

plugins: [dts({ insertTypesEntry: true })]

});

### Storybook básico

\*\*storybook/main.ts\*\*

import type { StorybookConfig } from '@storybook/react-vite';

const config: StorybookConfig = {

stories: ['../src/\*\*/\*.stories.@(js|jsx|ts|tsx)'],

addons: ['@storybook/addon-essentials'],

framework: '@storybook/react-vite'

};

export default config;

\*\*storybook/preview.ts\*\*

import type { Preview } from '@storybook/react';

const preview: Preview = { parameters: { layout: 'centered' } };

export default preview;

Documentación orientada a humanos

---------------------------------

### README.md

Debe incluir:

- Qué es la librería y para qué sirve.

- \*\*Instalación\*\* (dos opciones):

1) GitHub Packages:

npm config set @{{SCOPE:sin@}}:registry {{REGISTRY\_URL}}

npm i {{SCOPE}}/{{PKG\_NAME}}

2) Directo desde GitHub (alternativa rápida):

"dependencies": {

"{{SCOPE}}/{{PKG\_NAME}}": "git+ssh://git@github.com/{{ORG}}/{{REPO}}.git#v1.0.0"

}

- \*\*Uso básico\*\* con `ThemeProvider` y un componente.

- Lista de componentes con enlaces a sus stories y README\_AI.

### README\_AI.md (IA-first, corto)

- Propósito de la librería en 1 párrafo.

- Cómo integrarla en 6 pasos (instalar, envolver con ThemeProvider, importar, patrones de formularios, i18n local-first, testing).

- Normas de import (barrel `src/index.ts`).

- Tokens claves y convenciones (`intent: primary|secondary|danger`).

- Pitfalls comunes y soluciones.

- Sección “\*\*Si solo puedes leer 10 líneas\*\*” con 3 bullets críticos + 1 snippet.

Docs rápidas para adopción

--------------------------

### docs/quickstart.md

- Instalación (GitHub Packages o `git+ssh`).

- Registro de tokens (CI).

- Setup de `ThemeProvider` y `i18n` global.

- Import inicial y render.

- Comprobaciones (SSR/CSR, bundlers soportados).

### docs/integration-checklist.md

1. Acceso al registry (`NODE\_AUTH\_TOKEN`).

2. `ThemeProvider` global conectado a dark/light.

3. i18n global activo (lang + dict).

4. Import desde barrel (`{{SCOPE}}/{{PKG\_NAME}}`).

5. Verificar tree-shaking en tu bundler.

6. Aislar estilos (CSS Modules/Tokens).

7. Ejecutar `npm test` mínimo.

8. Verificar stories y ejemplos.

9. Confirmar tipos y `d.ts` presentes.

10. Versionado semantic-release listo.

Archivos AI‑friendly

--------------------

### ai/manifest.json (ejemplo)

{

"library": "{{SCOPE}}/{{PKG\_NAME}}",

"version": "0.1.0",

"components": [

{

"name": "Button",

"purpose": "Disparar acciones primarias/secundarias.",

"requiredProps": ["children"],

"commonProps": ["intent", "size", "disabled", "iconLeft", "iconRight", "onClick"],

"quickExample": "import { Button } from \"{{SCOPE}}/{{PKG\_NAME}}\"; <Button intent=\"primary\">Guardar</Button>",

"tokensUsed": ["color.primary", "radius.md", "spacing.2"],

"docs": "src/components/Button/README\_AI\_Component.md"

}

],

"globals": {

"provider": "ThemeProvider",

"i18n": "Local-first por defecto; si no existe la key local, usa global."

}

}

### ai/component-map.json

{ "Button": "src/components/Button/README\_AI\_Component.md" }

### ai/snippets/\*

- `button.install.md`: instalación rápida.

- `button.usage.md`: uso básico con props comunes.

- `form.pattern.md`: composición `TextField + Button` con submit.

Componentes de ejemplo

----------------------

> \*\*Estructura interna\*\* (carpetas y archivos detallados del componente) \*\*no va aquí\*\*.

> \*\*Usa el documento\*\* “Guía — Componente Avanzado React + TypeScript (local-first)” para crear cada componente (i18n local-first, tema dark/light, visibilidad opcional, hooks, provider, view, utils).

#### Button.tsx (ejemplo mínimo orientado a tokens + theme)

import React from 'react';

import { useTheme } from '../theme';

import { clsx } from '../utils/clsx';

export interface ButtonProps extends React.ButtonHTMLAttributes<HTMLButtonElement> {

intent?: 'primary' | 'secondary' | 'danger';

size?: 'sm' | 'md' | 'lg';

}

export const Button: React.FC<ButtonProps> = ({ intent='primary', size='md', className, ...rest }) => {

const { theme, tokens } = useTheme();

const cn = clsx('btn', `btn-${intent}`, `btn-${size}`, theme === 'dark' ? 'btn-dark' : 'btn-light', className);

return <button className={cn} style={{ borderRadius: tokens.radius.md }} {...rest} />;

};

#### Button.types.ts

export type ButtonIntent = 'primary' | 'secondary' | 'danger';

export type ButtonSize = 'sm' | 'md' | 'lg';

#### Button.stories.tsx

import type { Meta, StoryObj } from '@storybook/react';

import { Button } from './Button';

const meta: Meta<typeof Button> = { component: Button };

export default meta;

export const Primary: StoryObj<typeof Button> = { args: { children: 'Guardar', intent: 'primary' } };

export const Secondary: StoryObj<typeof Button> = { args: { children: 'Cancelar', intent: 'secondary' } };

export const Disabled: StoryObj<typeof Button> = { args: { children: 'Cargando', disabled: true } };

#### \_\_tests\_\_/Button.test.tsx

import { render, screen, fireEvent } from '@testing-library/react';

import { Button } from '../Button';

test('render & click', () => {

const onClick = vi.fn();

render(<Button onClick={onClick}>Guardar</Button>);

fireEvent.click(screen.getByText('Guardar'));

expect(onClick).toHaveBeenCalled();

});

Barrel exports

--------------

\*\*src/index.ts\*\*

export \* from './components/Button';

export \* from './theme';

export \* from './hooks';

export \* from './utils';

Theming (tokens + provider)

---------------------------

\*\*src/theme/tokens.ts\*\*

export const tokens = {

color: {

fg: { light: '#111827', dark: '#f9fafb' },

bg: { light: '#ffffff', dark: '#111827' },

primary: '#2563eb'

},

radius: { md: 8 },

spacing: { 2: 8 }

};

\*\*src/theme/ThemeProvider.tsx\*\*

import React, { createContext, useContext } from 'react';

import { tokens } from './tokens';

const Ctx = createContext<{ theme: 'light'|'dark'; tokens: typeof tokens }>({ theme: 'light', tokens });

export const ThemeProvider: React.FC<{ theme?: 'light'|'dark'; children: React.ReactNode }> = ({ theme='light', children }) => {

return <Ctx.Provider value={{ theme, tokens }}>{children}</Ctx.Provider>;

};

export const useTheme = () => useContext(Ctx);

Tooling

-------

- \*\*Vitest\*\* + Testing Library.

- \*\*ESLint\*\* + \*\*Prettier\*\* (con configs mínimas).

- \*\*Storybook\*\* (dev docs).

- \*\*Vite\*\* (bundle ESM + `d.ts`).

Consumo desde otros repos

-------------------------

### Opción A (recomendada): GitHub Packages

npm config set @{{SCOPE:sin@}}:registry {{REGISTRY\_URL}}

npm i {{SCOPE}}/{{PKG\_NAME}}

### Opción B: dependencia `git+ssh` fijando tag

"dependencies": {

"{{SCOPE}}/{{PKG\_NAME}}": "git+ssh://git@github.com/{{ORG}}/{{REPO}}.git#v1.0.0"

}

Notas finales

-------------

- i18n \*\*local-first\*\* por defecto.

- Respuesta a \*\*tema global\*\* dark/light por tokens + provider.

- Estructura de \*\*componentes\*\*: ver documento companion “Guía — Componente Avanzado React + TypeScript (local-first)”.