02 — Bootstrap

02 — Bootstrap

> Do repositório vazio até `npm run dev` servindo a aplicação. Siga na ordem; cada passo assume o anterior. Ao final, o checklist "Fundação" de 00-INDICE.md deve estar todo marcado.

1. Node

O projeto exige Node ≥ 24 (última LTS). Fixe a versão no repositório para que os git hooks consigam validá-la sem depender do shell do dev.

```
// .node-version
24.19.0
```

Com fnm:

```
fnm install 24 && fnm use
```

> **Armadilha conhecida.** Se o `fnm` não estiver avaliado no perfil do shell (`fnm env`), o Node ativo pode ser uma versão antiga e o `setEnvironment.js` falha em `process.loadEnvFile`. GUIs de Git tipicamente não herdam o perfil — por isso o `pre-commit` valida a versão explicitamente (ver 14).

2. Esqueleto

```
npm create vite@latest . -- --template react-ts
```

Depois **remova** o que o template traz e não usamos: `src/App.css`, `src/index.css`, `src/assets/`. **Mantenha** `eslint.config.js` — o próprio scaffold do Vite já gera flat config, e é isso que este guia usa (ver 14); você vai reescrever o conteúdo, não trocar de formato.

3. Dependências

Sem `@latest` em nada. `npm i` sem versão instala o mais novo do registro, o que pode não ter sido verificado ainda contra este guia — hoje só o TypeScript está deliberadamente atrás do latest (01). Pin explícito em tudo:

```
npm i @curio/client@^1.4.2 @emotion/react@^11.14.0 @emotion/styled@^11.14.1 @hookform/resolvers@^5.9.1 @mui/icons-material@^9.3.1 @mui/material@^9.3.1 @mui/x-date-pickers@^9.11.0 @tanstack/react-query@^5.101.4 @tanstack/react-query-devtools@^5.101.4 date-fns@^4.4.0 react@^19.2.0 react-dom@^19.2.0 react-hook-form@^7.85.0 react-router-dom@^7.18.2 zod@^4.4.3
```

```
npm i -D @eslint/js@^10.0.1 @types/node@^26.2.0 @types/react@^19.2.2 @types/react-dom@^19.2.4 @typescript-eslint/eslint-plugin@^8.67.0 @typescript-eslint/parser@^8.67.0 @vitejs/plugin-react@^6.0.5 eslint@^10.8.1 eslint-config-prettier@^10.1.8 eslint-plugin-react-hooks@^7.1.1 eslint-plugin-react-refresh@^0.5.4 globals@^17.11.0 husky@^9.1.7 lint-staged@^17.3.0 prettier@^3.9.6 typescript@^6.0.3 vite@^8.2.1 vite-plugin-checker@^0.14.5
```

Versões exatas em 01-STACK-E-DECISOES.md — essa tabela é a fonte de verdade; se as duas divergirem, ela vence.

> `@types/node@^26` **com Node** `>=24` **em runtime é uma folga deliberada.** A linha 26.x dos tipos é a mais recente disponível hoje; a linha 24.x parou em `24.13.3` (sem novas patches). Não houve conflito de `tsc`/build ao testar essa combinação de verdade, mas se aparecer algum tipo de API que não existe no Node 24 real, prenda de volta em `^24`.

4. `package.json`

```
{
  "name": "nome-do-projeto",
  "version": "1.0.0",
  "private": true,
  "type": "module",
  "scripts": {
    "env:dev": "node ./setEnvironment.js dev",    "env:homolog": "node ./setEnvironment.js
homolog",
    "env:prod": "node ./setEnvironment.js prod",
    "dev": "npm run env:dev && vite",
    "start": "npm run env:dev && vite",
    "start:homolog": "npm run env:homolog && vite",    "start:prod": "npm run env:prod &&
vite",
    "build": "npm run lint && npm run env:prod && tsc && vite build",    "build:homolog": "npm
run env:homolog && tsc && vite build",
    "preview": "vite preview",    "lint": "eslint . --report-unused-disable-directives --max-
warnings 0",
    "format": "prettier --write \"src/**/*.{ts,tsx,css,json,md}\"",    "format:check":
"prettier --check \"src/**/*.{ts,tsx,css,json,md}\"",
    "prepare": "husky"
  },
  "lint-staged": {
    "src/**/*.{ts,tsx}": [      "eslint --fix --report-unused-disable-directives --max-warnings
0",
      "prettier --write"
    ],
```

```
    "**/*.{json,css,scss,md,html,yml,yaml}": ["prettier --write"]
  },
  "engines": { "node": ">=24.0.0" }
}
```

> `prepare` **depende do layout do repositório.** Acima está a forma para um repositório de módulo único (projeto web na raiz). Se o web for submódulo de um monorepo, o script muda de forma — algo como `"prepare": "cd ../.. && husky caminho/do/modulo/.husky"`, apontando para a raiz real do repositório git. Se o projeto novo for um módulo dentro de um monorepo, replique essa forma — ver 14.

5. `vite.config.ts`

```
import { defineConfig } from "vite";
import react from "@vitejs/plugin-react";
import { resolve } from "path";
import checker from "vite-plugin-checker";
export default defineConfig({
  plugins: [
    react(),
    checker({
      typescript: true,
      overlay: { initialIsOpen: false, position: "tl" },
      terminal: true
    })
  ],
  define: {
    // curio espera globals de node; browser nao tem
    global: "globalThis",
    "process.env": {}
  },
  resolve: {
```

```
    alias: {
      "@": resolve(import.meta.dirname, "./src"),      "@components":
resolve(import.meta.dirname, "./src/components"),      "@pages": resolve(import.meta.dirname,
"./src/pages"),
      "@hooks": resolve(import.meta.dirname, "./src/hooks"),      "@utils":
resolve(import.meta.dirname, "./src/utils"),      "@types": resolve(import.meta.dirname,
"./src/types"),
      "@api": resolve(import.meta.dirname, "./src/api"),      "@context":
resolve(import.meta.dirname, "./src/context"),      "@theme": resolve(import.meta.dirname,
"./src/theme")
    }
  },
  server: { port: 5000, open: true },
  build: { outDir: "dist", sourcemap: true }
});
```

O bloco `define` **não é opcional**: `@curio/client` referencia `global` e `process.env`, que não existem no browser. Sem ele a aplicação quebra em runtime no primeiro request.

6. `tsconfig.json`

```
{
  "compilerOptions": {
    "types": ["vite/client"],
    "target": "ES2020",
    "useDefineForClassFields": true,
    "lib": ["ES2020", "DOM", "DOM.Iterable"],
    "module": "ESNext",
    "skipLibCheck": true,
    "moduleResolution": "bundler",
    "allowImportingTsExtensions": true,
    "resolveJsonModule": true,
    "isolatedModules": true,
    "noEmit": true,
```

```
    "jsx": "react-jsx",
    "strict": true,
    "noUnusedLocals": true,
    "noUnusedParameters": true,
    "noFallthroughCasesInSwitch": true,
    "paths": {
      "@/*": ["./src/*"],
      "@components/*": ["./src/components/*"],
      "@pages/*": ["./src/pages/*"],
      "@hooks/*": ["./src/hooks/*"],
      "@utils/*": ["./src/utils/*"],
      "@types/*": ["./src/types/*"],
      "@api/*": ["./src/api/*"],
      "@context/*": ["./src/context/*"],
      "@theme/*": ["./src/theme/*"]
    }
  },
  "include": ["src"],
  "references": [{ "path": "./tsconfig.node.json" }]
}
```

> **Regra:** todo alias novo entra nos **dois** arquivos. Só no `vite.config.ts` → o build passa e o editor reclama. Só no `tsconfig.json` → o editor aceita e o build quebra.

7. Configuração de ambiente

Crie `setEnvironment.js`, `config/*.json` e `.env.example` conforme 04-AMBIENTES-E-CONFIG.md. Sem isso, `npm run dev` falha no primeiro passo.

8. `.gitignore`

```
```

```
# .gitignore
node_modules
dist
dist-ssr
*.local
.env
# gerado por setEnvironment.js — nao versionar
public/config.json
.vscode/*
!.vscode/extensions.json
.idea
.DS_Store
*.suo
*.ntvs*
*.njsproj
*.sln
*.sw?
# Harness (Claude Code) — estado local, nao versionado
harness/state/*.json
harness/state/*.md
!harness/state/_examples/
harness/handoffs/*.md
!harness/handoffs/_examples/
harness/user_preferences.md
```

9. Ponto de entrada

`index.html`, `src/main.tsx` e `src/App.tsx` conforme 03-ESTRUTURA-DE-PASTAS.md.

10. Hooks reutilizáveis

Copie os doze hooks/módulos de 21-CATALOGO-DE-HOOKS.md para `src/hooks/`, mais `src/utils/useCaseTriggersLogger.ts`, e crie o `src/hooks/index.ts`.

Faça isso **agora**, não na primeira tela: `useCurioMutation` e `useUseCaseControls` são a única forma suportada de chamar um caso de uso (06), e `useAuthQuery` é exigido pelo `AuthProvider` (05).

11. Tela de exemplo (placeholder)

Antes de existir qualquer contrato real com o backend, crie uma tela `Exemplo` seguindo exatamente a estrutura de 17-PRIMEIRA-TELA.md, mas com **todo id de caso de uso e nome de RM como placeholder explícito** (`"SUBSTITUA_USE_CASE_ID"`, `"RM_SUBSTITUA_SALVAR"`, ...) — nunca invente um valor plausível que pareça real.

Isso é obrigatório, não opcional, por dois motivos:

- É a única forma de provar, com `tsc` + `lint` + `build` reais, que a cadeia completa (`service/constants.ts` → `service/interfaces.ts` → `service/hooks.ts` → `Page.tsx` → `UseCaseManager` → `paths.ts`/`routes.ts`/`menuTree.ts`) compila e roteia antes de qualquer feature de negócio existir.
- Dá ao time um exemplo executável para copiar, em vez de reconstruir o padrão de memória a cada tela nova.

Nomeie a entidade de forma que **não possa ser confundida com domínio real** — `Exemplo`, não uma entidade do produto. Comente no topo do arquivo que é placeholder e que deve ser removido/substituído quando a primeira tela de negócio for criada. Registre a rota e o item de menu normalmente (09, 10) — a tela precisa ser navegável e verificável no browser, não só compilar.

12. Qualidade e harness

- ESLint, Prettier, husky → 14
- `harness/` e `init.sh` → 18 e 19

13. Verificar

```
```

```
npm run lint && npx tsc --noEmit && npm run dev
```

Os três precisam passar. Se `npm run dev` abrir a página mas o console mostrar erro de `global is not defined`, o bloco `define` do passo 5 está faltando.

Revision #2
Created Thu, Aug 20, 2026 4:45 PM by Geraldo Barbosa
Updated Tue, Aug 25, 2026 4:50 PM by Geraldo Barbosa