



? Nome do componente: Header




✓ ++ \src\components\UI\Header\index.tsx

✓ ++ \src\components\UI\Header\Header.stories.tsx


✓ ++ \src\components\UI\Header\Header.test.tsx

✓  Header



 Header.stories.tsx

 Header.test.tsx

 index.tsx

CRIANDO CLI NO REACT.JS



Isaac Gomes

TS

CLI - COMMAND LINE INTERFACE

a criação de arquivos via
linha de comando é algo comum em
frameworks, pois, fica fácil de
aplicar padrões e acelerar o
Desenvolvimento



Isaac Gomes

TS

EXEMPLO



```
nest g resource
```

Quando trabalhamos com **Nest.js** podemos gerar arquivos via terminal simplificando a **criação de arquivos**



Isaac Gomes



1 - instalar o plop e ts-node

```
$ npm i plop
```

```
added 168 packages, and audited 386 packages in 22s
```

```
90 packages are looking for funding  
  run `npm fund` for details
```

```
found 0 vulnerabilities
```

```
$ npm i ts-node
```

```
added 14 packages, and audited 400 packages in 8s
```

```
90 packages are looking for funding  
  run `npm fund` for details
```

```
found 0 vulnerabilities
```



Isaac Gomes

TS

2 - criar generates



```
✓ generate
  Component.tsx
  stories.tsx
  test.tsx
```



Isaac Gomes

TS

2 - criar generates

COMPONENT

```
Component.ts

type {{name}}Props = {}

export const {{name}} = ({}: {{name}}Props) => {
  return (
    <div></div>
  );
}
```



Isaac Gomes

TS

2 - criar generates

STORIES

```
stories.ts

import type { Meta, StoryObj } from "@storybook/react"
import { {{name}} } from "."

export type Story = StoryObj<typeof {{name}}>

const meta: Meta<typeof {{name}}> = {
  component: {{name}},
  parameters: {
    layout: "centered",
  },
}

export default meta

export const Default: Story = {}
```



Isaac Gomes

TS

2 - criar generates

TEST

test.ts

```
import { {{name}} } from "."

describe(`<{{name}} />`, () => {
  test("", () => {})
})
```



Isaac Gomes

TS

3 - adicionar o comando de geração de arquivos nos scripts do projeto

scripts.ts

```
"scripts": {  
  "dev": "vite",  
  "build": "tsc && vite build",  
  "preview": "vite preview",  
  "generate": "node --loader ts-node/esm node_modules/plop/bin/plop"  
},
```



Isaac Gomes

TS

4 - configurar gerações de arquivos

```
plopfile.ts

export default function (plop: NodePlopAPI) {
  plop.setGenerator("component", {
    description: "Gerador de componentes React",
    prompts: [{
      type: "input",
      name: "name",
      message: "Nome do componente:",
    }],
    actions: [{
      type: "add",
      path: "src/components/UI/{{name}}/index.tsx",
      templateFile: "generate/Component.tsx",
    }, {
      type: "add",
      path: "src/components/UI/{{name}}/{{name}}.stories.tsx",
      templateFile: "generate/stories.tsx",
    }, {
      type: "add",
      path: "src/components/UI/{{name}}/{{name}}.test.tsx",
      templateFile: "generate/test.tsx",
    }],
  })
}
```



Isaac Gomes

TS

5 - agora é só testar

Componentes sendo gerados

```
$ npm run generate

> criando_cli_react@0.0.0 generate
> node --loader ts-node/esm node_modules/plop/bin/plop.js

(node:5704) ExperimentalWarning: Custom ESM Loaders
(Use `node --trace-warnings ...` to show where the warning
? Nome do componente: Header
✓ ++ \src\components\UI\Header\index.tsx
✓ ++ \src\components\UI\Header\Header.stories.tsx
✓ ++ \src\components\UI\Header\Header.test.tsx
```



Isaac Gomes

TS

5 - agora é só testar

Resultado

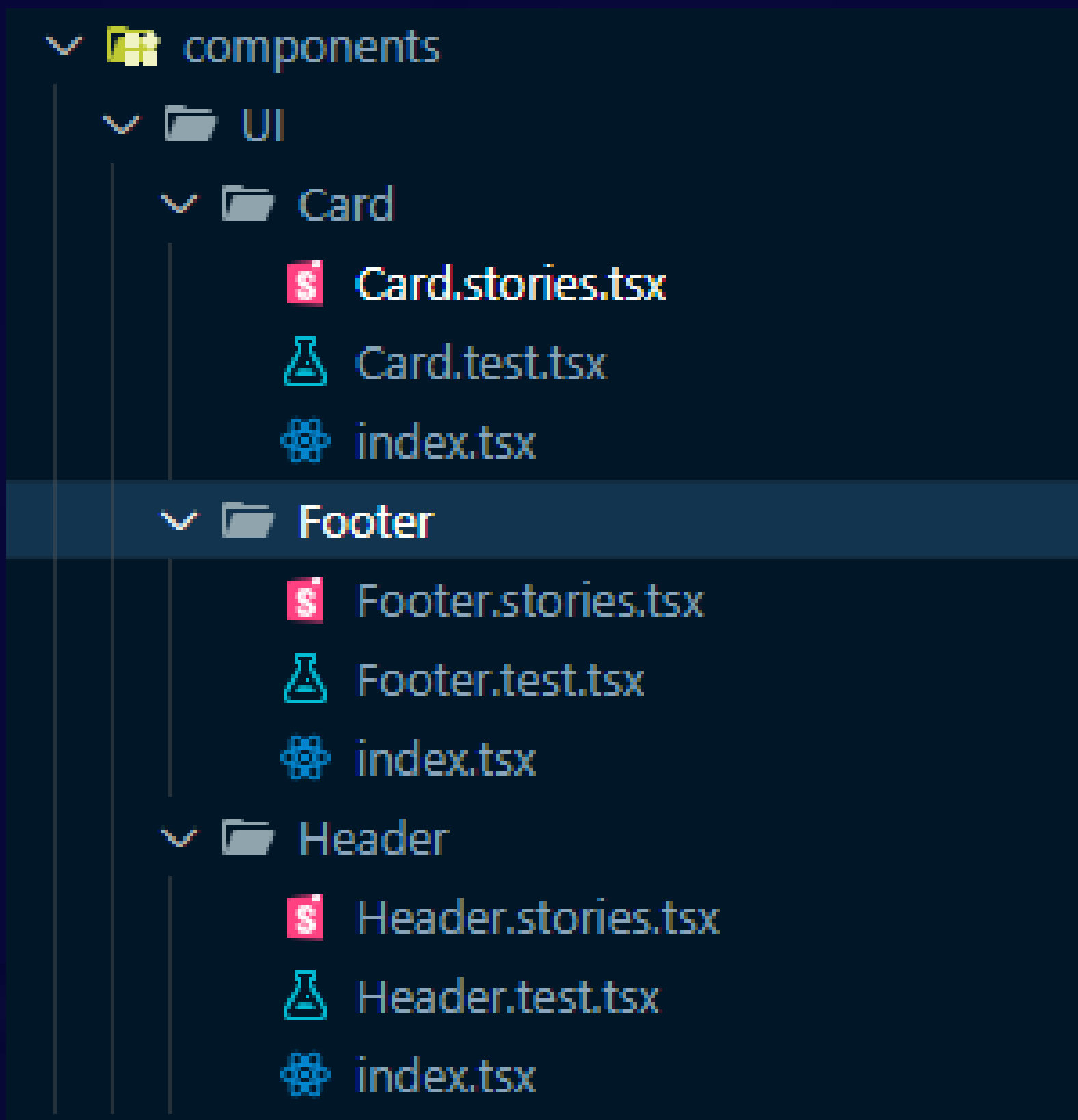
```
▼ components
  ▼ UI
    ▼ Header
      $ Header.stories.tsx
      ⚗ Header.test.tsx
      ⚙ index.tsx
```



Isaac Gomes

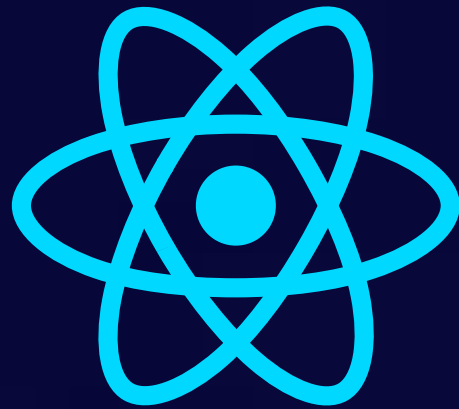
TS

agora podemos gerar nossas estruturas via **linha de Comando**



Isaac Gomes

TS



para o exemplo usei a estrutura de
Componentes, porem, podemos
adicionar mais como possibilidades
como **pages, services, ...etc**



Isaac Gomes

TS

Gostou?



Curta



Compartilhe



Salve



Isaac Gomes

TS