Implementando gráficos com ShadCN

Link Notion: https://cherry-client-b8f.notion.site/Implementando-gr-ficos-com-ShadCN-276911d84e0d80f7bb13fb74f53ad67b?pvs=73

Existem diversas bibliotecas que trabalham com a implementação de gráficos como:

https://recharts.org/en-US (base por trás do ShadCN)

https://www.chartjs.org/

Utilizaremos os recursos de gráficos oferecido pelo ShadCN:

https://ui.shadcn.com/charts/area

O que é o ShadCN?

O ShadCN é uma biblioteca para criar **componentes UI acessíveis** e **estilizados** com base no TailwindCSS. Ele facilita a construção de layouts modernos, como dashboards, sem precisar reinventar a roda.

Como instalar?

Na documentação temo o passo a passo, porém caso o projeto não utilize TypeScript, algumas alterações são necessárias.

https://ui.shadcn.com/docs/installation/vite

1º Passo

Você deve instalar o Tailwind utilizando o comando:

```
npm install tailwindcss @tailwindcss/vite
```

2º Passo

Remova todo o conteúdo do arquivo index.css padrão do Vite e substitua por:

```
@import "tailwindcss";
```

3º Passo

Crie um arquivo chamado jsconfig.json e cole o código abaixo:

```
{
  "references": [],
  "compilerOptions": {
    "baseUrl": ".",
    "paths": {
      "@/*": ["./src/*"]
    }
  }
}
```

4º Passo

Substitua o conteúdo do arquivo viteconfig.js pelo código abaixo:

```
import path from "path"
import tailwindcss from "@tailwindcss/vite"
import react from "@vitejs/plugin-react"
import { defineConfig } from "vite"
```

```
// https://vite.dev/config/
export default defineConfig({
  plugins: [react(), tailwindcss()],
  resolve: {
    alias: {
      "@": path.resolve(__dirname, "./src"),
    },
  },
})
```

5º Passo

Execute no terminal o comando:

```
npx shadcn@latest init
```

Você será perguntado sobre qual cor quer utilizar como base dos componentes. Utilizaremos no exemplo de aula a opção **Neutral**.

```
Which color would you like to use as base color? › Neutral
```

6º Passo

Para se certificar que está tudo certo, a documentação sugere a criação de um botão simples, executando o comando:

```
npx shadcn@latest add button
```

E alterando o conteúdo do componente App.jsx para:

```
import { Button } from "@/components/ui/button"

function App() {
  return (
    <div className="flex min-h-svh flex-col items-center justify-center">
      <Button>Click me</Button>
    </div>
  )
}

export default App
```

Utilizando gráficos

Para utilizar os gráficos, precisamos primeiramente importar o Recharts e suas dependências, com o comando:

```
npx shadcn@latest add chart
```

A partir desse momento você consegue configurar e construir os gráficos de forma personalizada, seguindo passo a passo os requisitos (https://ui.shadcn.com/docs/components/chart) ou copiando gráficos pré-configurados (https://ui.shadcn.com/charts/area).

Abaixo um exemplo de um gráfico em barras:

```
import { TrendingUp } from "lucide-react"
import { Bar, BarChart, CartesianGrid, XAxis } from "recharts"

import {
```

```
  Card,
  CardContent,
  CardDescription,
  CardFooter,
  CardHeader,
  CardTitle,
} from "@/components/ui/card"
import {
  ChartConfig,
  ChartContainer,
  ChartTooltip,
  ChartTooltipContent,
} from "@/components/ui/chart"

export const description = "A bar chart"

const chartData = [
  { month: "January", desktop: 186 },
  { month: "February", desktop: 305 },
  { month: "March", desktop: 237 },
  { month: "April", desktop: 73 },
  { month: "May", desktop: 209 },
  { month: "June", desktop: 214 },
]

const chartConfig = {
  desktop: {
    label: "Desktop",
    color: "var(--chart-1)",
  },
}

export function ChartBarDefault() {
  return (
    <Card>
      <CardHeader>
        <CardTitle>Bar Chart</CardTitle>
        <CardDescription>January - June 2024</CardDescription>
      </CardHeader>
      <CardContent>
        <ChartContainer config={chartConfig}>
          <BarChart accessibilityLayer data={chartData}>
            <CartesianGrid vertical={false} />
            <XAxis
              dataKey="month"
              tickLine={false}
              tickMargin={10}
              axisLine={false}
              tickFormatter={(value) => value.slice(0, 3)}
            />
            <ChartTooltip
              cursor={false}
              content={<ChartTooltipContent hideLabel />}
            />
            <Bar dataKey="desktop" fill="var(--color-desktop)" radius={8} />
          </BarChart>
        </ChartContainer>
      </CardContent>
```

```
      <CardFooter className="flex-col items-start gap-2 text-sm">
        <div className="flex gap-2 leading-none font-medium">
          Trending up by 5.2% this month <TrendingUp className="h-4 w-4" />
        </div>
        <div className="text-muted-foreground leading-none">
          Showing total visitors for the last 6 months
        </div>
      </CardFooter>
    </Card>
  )
}
```

Quebrando o componente em partes

▼ Importações

```
import { TrendingUp } from "lucide-react"
import { Bar, BarChart, CartesianGrid, XAxis } from "recharts"

import {
  Card,
  CardContent,
  CardDescription,
  CardFooter,
  CardHeader,
  CardTitle,
} from "@/components/ui/card"

import {
  ChartConfig,
  ChartContainer,
  ChartTooltip,
  ChartTooltipContent,
} from "@/components/ui/chart"
```

- `lucide-react` → ícone usado no footer (`TrendingUp`). Link para a biblioteca http://lucide.dev/
- `recharts` → componentes do gráfico de barras:
  - `BarChart` → container do gráfico
  - `Bar` → as barras
  - `CartesianGrid` → grid de fundo
  - `XAxis` → eixo X (meses, labels)
- `shadcn/ui` → componente visual (Card, Header, Footer).
- `ChartContainer`, `ChartTooltip`, `ChartTooltipContent` → wrappers e tooltip customizados do projeto.

▼ Dados do gráfico

```
const chartData = [
  { month: "January", desktop: 186 },
  { month: "February", desktop: 305 },
  { month: "March", desktop: 237 },
  { month: "April", desktop: 73 },
  { month: "May", desktop: 209 },
  { month: "June", desktop: 214 },
]
```

- Array de objetos → cada objeto é uma **linha no gráfico**.
- `month` → label do eixo X
- `desktop` → valor que define a **altura da barra**.

▼ Configurações do gráfico

```
const chartConfig = {
  desktop: {
    label: "Desktop",
    color: "var(--chart-1)",
  },
}
```

- Configuração de cores e labels para o gráfico.
- `desktop` → corresponde à chave no `chartData` que será usada pelo `Bar`.

▼ Estrutura do componente

```
<Card>
  <CardHeader>
    <CardTitle>Bar Chart</CardTitle>
    <CardDescription>January - June 2024</CardDescription>
  </CardHeader>
  <CardContent>
    <ChartContainer config={chartConfig}>
      <BarChart accessibilityLayer data={chartData}>
        ...
      </BarChart>
    </ChartContainer>
  </CardContent>
  <CardFooter>...</CardFooter>
</Card>
```

- **Card** → container visual.
- **CardHeader** → título e descrição do gráfico.
- **CardContent** → contém o gráfico.
- **ChartContainer** → wrapper do projeto, aplica cores, legendas e padding.
- **BarChart** → componente principal do Recharts. Recebe:
  - `data` → dados do gráfico
  - `accessibilityLayer` → recurso de acessibilidade (labels para leitores de tela).

▼ Componentes internos

CartesianGrid

```
<CartesianGrid vertical={false} />
```

- Mostra a grade horizontal do gráfico.
- `vertical={false}` → remove linhas verticais.

XAxis

```
<XAxis
  dataKey="month"
  tickLine={false}
```

```
  tickMargin={10}
  axisLine={false}
  tickFormatter={(value) => value.slice(0, 3)}
/>
```

- `dataKey="month"` → pega o valor da chave `month` para o eixo X.
- `tickLine={false}` → remove o pequeno traço que aparece nas linhas, chamado tick.
- `tickMargin={10}` → distância do tick até o eixo.
- `axisLine={false}` → remove linha do eixo X.
- `tickFormatter` → formata o label, nesse caso pega apenas os **3 primeiros caracteres do mês** (Jan, Feb, Mar...).

ChartTooltip

```
<ChartTooltip
  cursor={false}
  content={<ChartTooltipContent hideLabel />}
/>
```

- Tooltip customizado do projeto (shadcn/ui).
- `cursor={false}` → remove efeito de hover da barra, que coloca uma sombra atrás da barra.
- `ChartTooltipContent` → componente visual do tooltip.

Bar

```
<Bar dataKey="desktop" fill="var(--color-desktop)" radius={8} />
```

- `dataKey="desktop"` → pega o valor do objeto para definir **altura da barra**.
- `fill` → cor da barra.
- `radius={8}` → bordas arredondadas da barra.

▼ Footer

```
<CardFooter className="flex-col items-start gap-2 text-sm">
  <div className="flex gap-2 leading-none font-medium">
    Trending up by 5.2% this month <TrendingUp className="h-4 w-4" />
  </div>
  <div className="text-muted-foreground leading-none">
    Showing total visitors for the last 6 months
  </div>
</CardFooter>
```

- Área inferior do card.
- Mostra **informações adicionais**, ícone e texto.
- `TrendingUp` → ícone da biblioteca lucide.