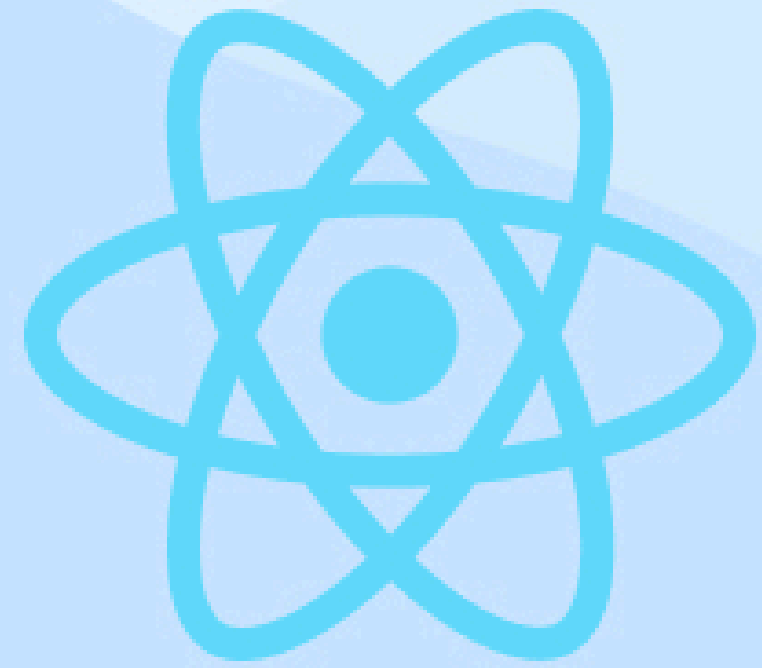
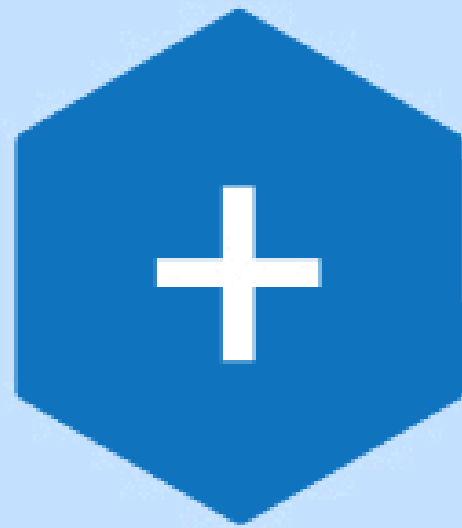


REACT COM  
TYPESCRIPT



**Typescript**


**ReactJS**

# IMPORTAÇÃO E EXPORTAÇÃO DE MÓDULOS NO TYPESCRIPT

No TypeScript, podemos dividir o código em módulos para melhorar a organização e reutilização.

## Exportação de Módulos

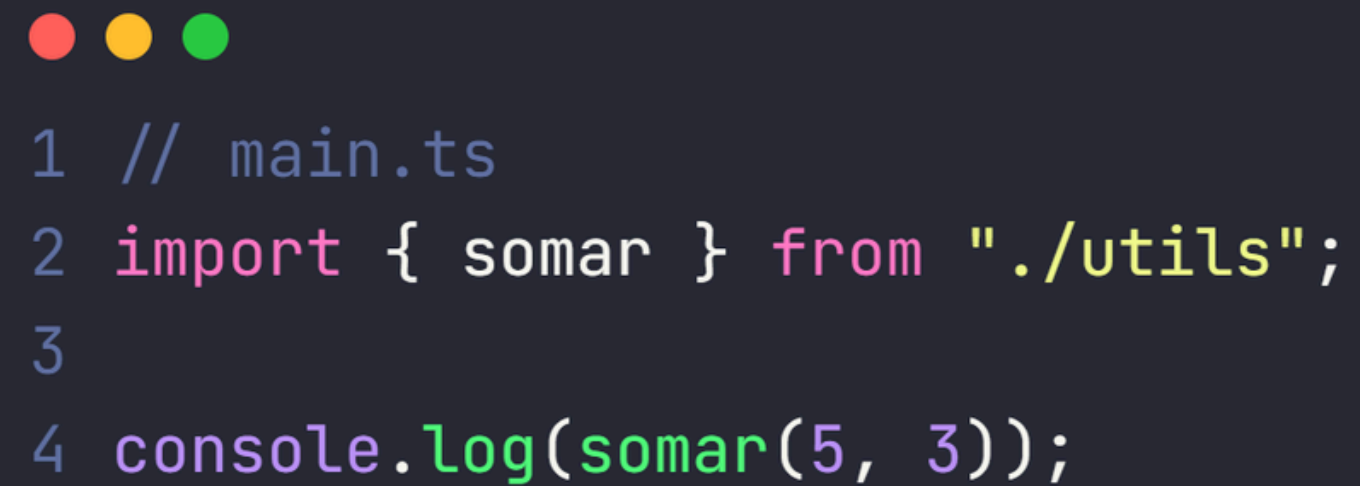
Para exportar uma função, variável ou classe, usamos a palavra-chave `export`:



```
1 // utils.ts
2 export function somar(a: number, b: number): number {
3     return a + b;
4 }
```

## Importação de Módulos

Para importar algo de um módulo:



```
1 // main.ts
2 import { somar } from "./utils";
3
4 console.log(somar(5, 3));
```

## Exportação Padrão (Default Export)

Podemos exportar algo como padrão:



```
1 // calculadora.ts
2 export default function multiplicar(a: number, b: number): number {
3     return a * b;
4 }
```

## Exportação Padrão (Default Export)

Podemos exportar algo como padrão:

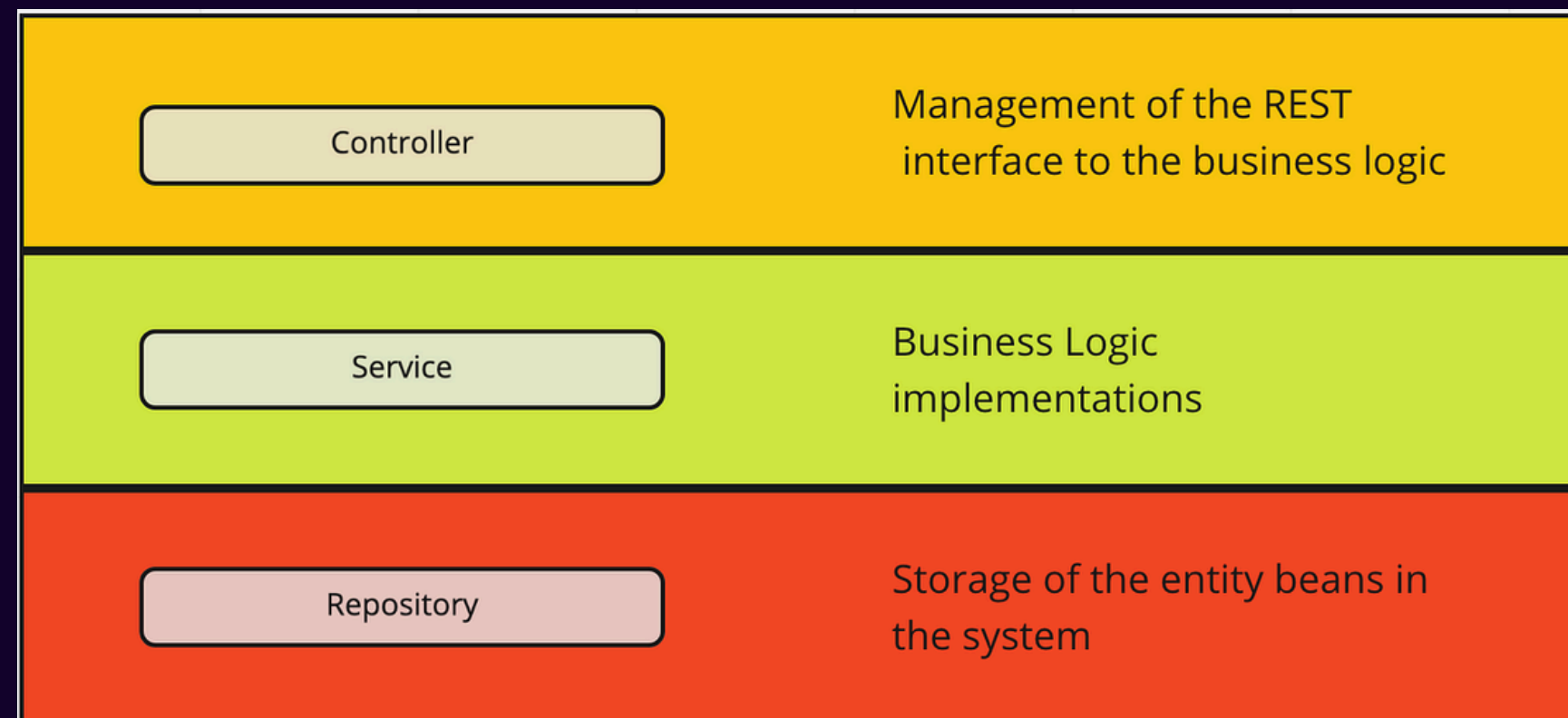


```
1 // main.ts
2 import multiplicar from "./calculadora";
3 console.log(multiplicar(4, 2));
```

# SEPARAÇÃO DE CÓDIGO EM ARQUIVOS DISTINTOS

A separação do código melhora a manutenção do projeto. Um exemplo seria dividir um sistema de usuários em diferentes arquivos:

- models/User.ts
- services/UserService.ts
- controllers/UserController.ts



# CRIANDO UM SISTEMA DE TIPOS REUTILIZÁVEIS

Criamos um arquivo types.ts para centralizar os tipos utilizados no projeto:



```
1 // types.ts
2 export interface Funcionario {
3   id: number;
4   nome: string;
5   cargo: string;
6   salario: number;
7   dataAdmissao: string;
8 }
```

Podemos reutilizar esse tipo em qualquer arquivo:

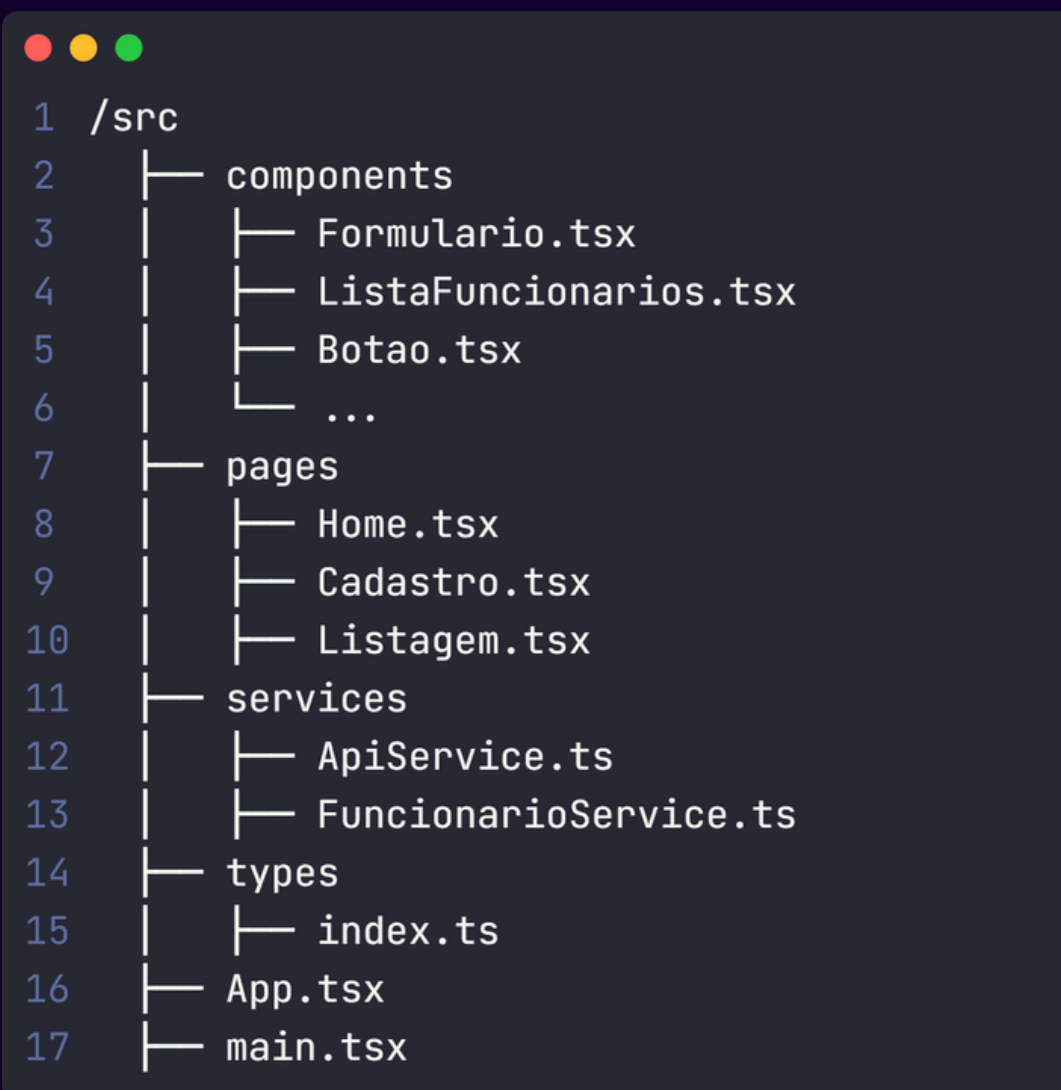


```
1 // services/FuncionarioService.ts
2 import { Funcionario } from "../types";
3
4 const funcionarios: Funcionario[] = [];
5
6 export function adicionarFuncionario(funcionario: Funcionario) {
7     funcionarios.push(funcionario);
8 }
```



# ORGANIZAÇÃO DO CÓDIGO EM UM PROJETO REAL

Para manter o código organizado, usamos a estrutura de pastas a seguir:

A terminal window with a dark background and light blue text. It shows a file tree structure for a project. The root directory is /src. Under /src, there are four subdirectories: components, pages, services, and types. Each subdirectory contains several files. The components directory contains Formulario.tsx, ListaFuncionarios.tsx, Botao.tsx, and an ellipsis (...). The pages directory contains Home.tsx, Cadastro.tsx, and Listagem.tsx. The services directory contains ApiService.ts and FuncionarioService.ts. The types directory contains index.ts. Additionally, there are two files at the root of /src: App.tsx and main.tsx.

```
1 /src
2   | components
3   |   | Formulario.tsx
4   |   | ListaFuncionarios.tsx
5   |   | Botao.tsx
6   |   | ...
7   | pages
8   |   | Home.tsx
9   |   | Cadastro.tsx
10  |   | Listagem.tsx
11  | services
12  |   | ApiService.ts
13  |   | FuncionarioService.ts
14  | types
15  |   | index.ts
16  | App.tsx
17  | main.tsx
```

Essa estrutura separa componentes, páginas, serviços e tipos, facilitando o desenvolvimento.

# CONFIGURAÇÃO DO PROJETO REACT COM TYPESCRIPT

## Criar um Novo Projeto

Para iniciar um projeto React com TypeScript, usamos:



```
1 npx create-react-app meu-projeto --template typescript
```

Ou com Vite:



```
1 npm create vite@latest meu-projeto --template react-ts
```

# CONFIGURAR ESLINT E PRETTIER

Após criar o projeto, instalamos ESLint e Prettier:



```
1 npm install --save-dev eslint prettier eslint-config-prettier eslint-plugin-prettier  
  @typescript-eslint/parser @typescript-eslint/eslint-plugin
```

Criamos o arquivo `.eslintrc.json`:



```
1 {  
2   "extends": ["plugin:@typescript-eslint/recommended", "prettier"],  
3   "rules": {  
4     "prettier/prettier": "error"  
5   }  
6 }
```

Criamos o arquivo `.prettierrc`:



```
1 {  
2   "singleQuote": true,  
3   "semi": true  
4 }
```

Com essa aula, aprendemos a **modularizar** o código no TypeScript, **organizar** um projeto em arquivos distintos e **configurar** um projeto React com TypeScript eficientemente.

Agora estamos prontos para começar a implementar funcionalidades mais avançadas no nosso sistema!

THANK  
YOU

@wallace027dev