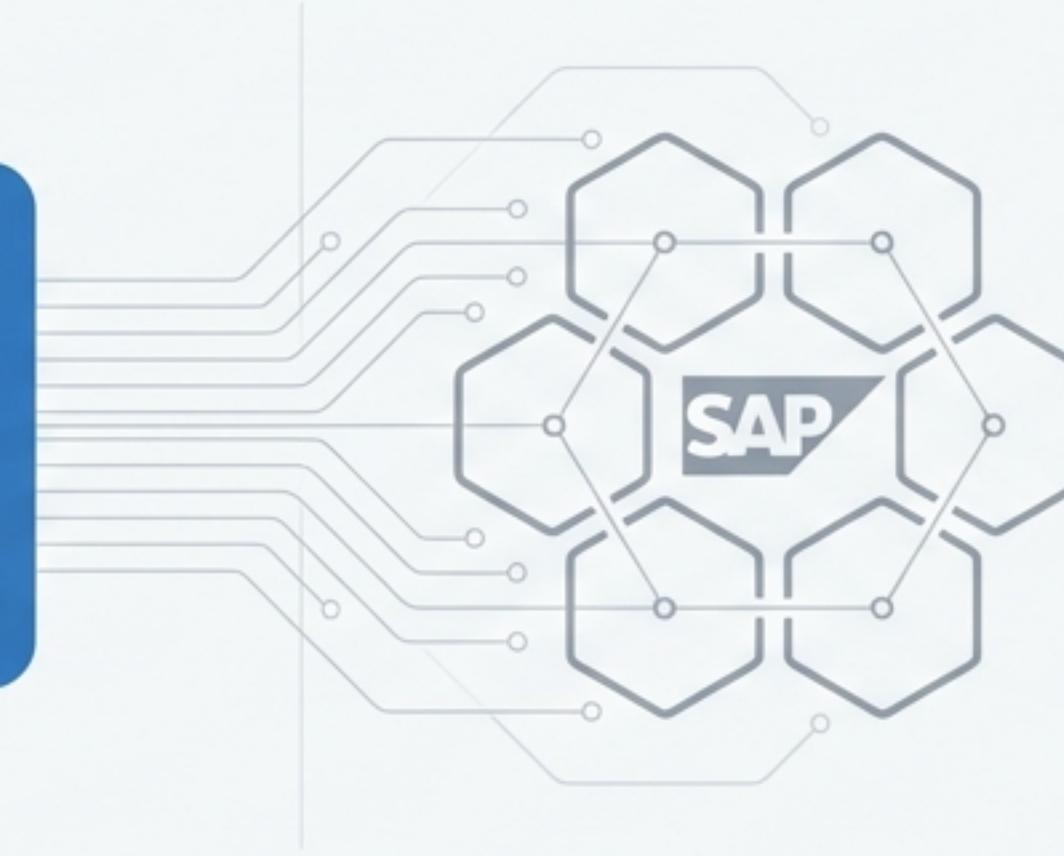


Dominando TypeScript no CAP Node.js

Um Guia Completo do Setup à Produção



Um guia visual e prático para integrar TypeScript de forma eficaz em seus projetos SAP Cloud Application Programming Model (CAP).

A Jornada do Desenvolvedor com TypeScript

Mapeamos as fases essenciais do ciclo de vida do desenvolvimento, guiando você passo a passo.



1. Configuração: Habilitando o superpoder do TypeScript.



2. Desenvolvimento: Codificando com agilidade e feedback instantâneo.



3. Testes: Garantindo a qualidade com testes tipados.



4. Build: Preparando seu código para a produção.



5. Otimização: Elevando a produtividade com APIs e tipos automáticos.

Fase 1: A Configuração Inicial

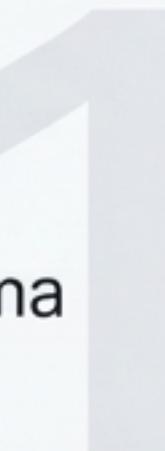
Como eu começo a usar TypeScript no meu projeto CAP?

Um processo simples de dois passos para habilitar o suporte a TypeScript.

Passo 1: Instale as dependências globais.

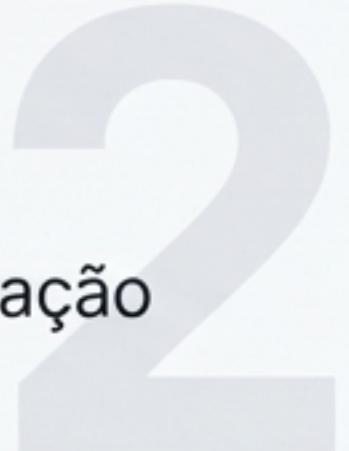
Ferramentas essenciais para o ecossistema TypeScript.

```
npm i -g typescript ts-node tsx
```



Passo 2: Adicione o "facet" TypeScript ao projeto.

Este comando automatiza a configuração inicial do seu projeto.

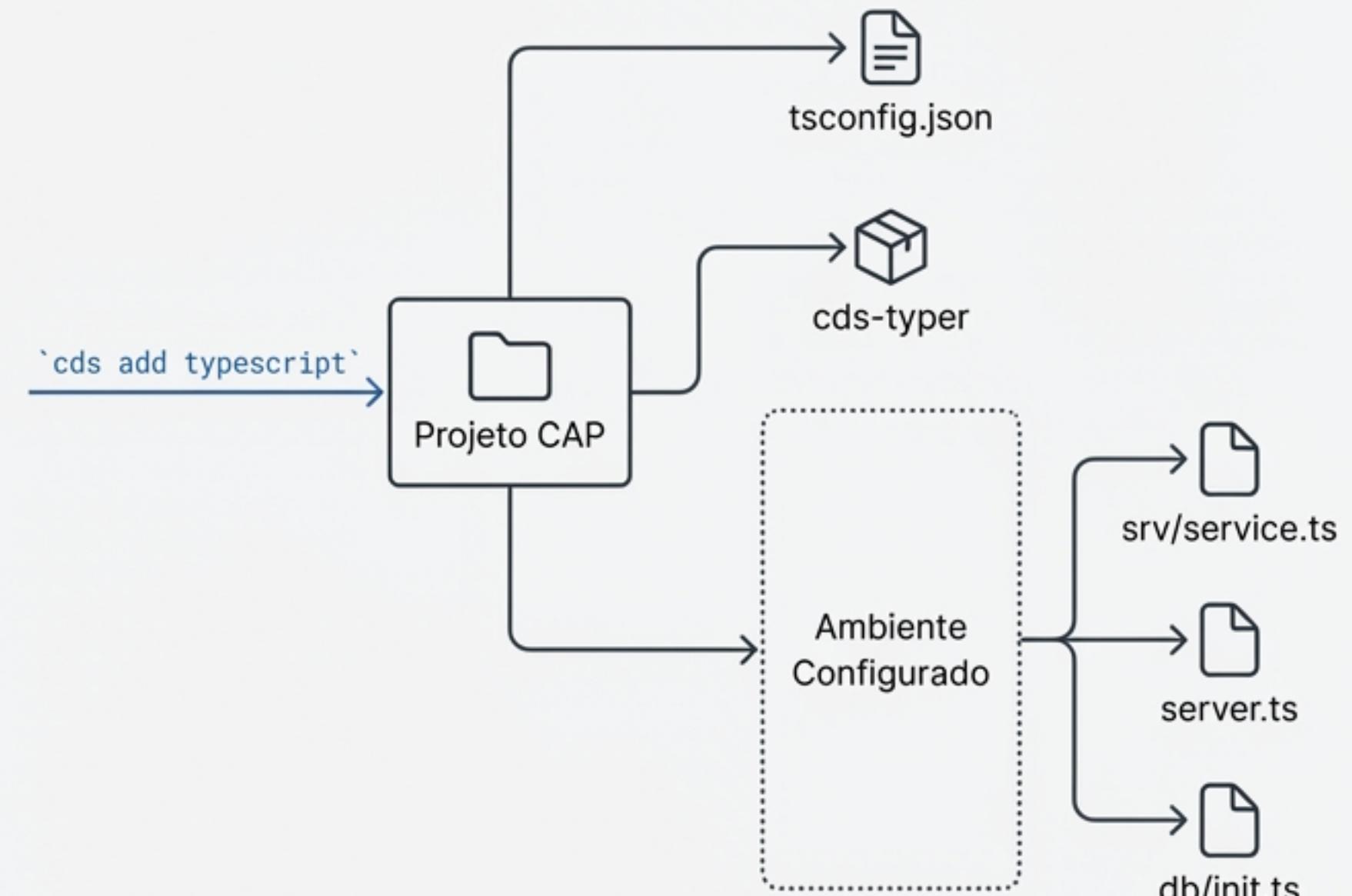


```
cds add typescript
```

O Que Acontece "Por Baixo dos Panos"?

O que o comando `cds add typescript` realmente faz?

- Cria `tsconfig.json`: Adiciona o arquivo de configuração essencial para o compilador TypeScript.
- Adiciona `cds-typer`: Instala `cds-typer` como uma dependência de desenvolvimento, uma ferramenta poderosa que veremos mais adiante.
- **Prepara o Ambiente:** Configura seu projeto para reconhecer arquivos `*.ts` em vez de `*.js` para:
 - Handlers de serviço (ex: `srv/service.ts`)
 - Servidores customizados (ex: `server.ts`)
 - Scripts de inicialização de banco de dados (ex: `db/init.ts`)



Fase 2: O Ciclo de Desenvolvimento

Como desenvolvo e vejo minhas alterações em tempo real?

A Via Rápida (Recomendado): `cds watch`

- Use o comando que você já conhece e ama.
- O CAP detecta automaticamente a presença de um `tsconfig.json`.
- Ele executa `cds-tsx` internamente para transpilação 'on-the-fly', sem a necessidade de passos manuais.



```
cap/sflight $ cds watch
Detected tsconfig.json. Running with tsx.
[cds] serving TravelService { impl: 'srv/travel-service.ts',
path: '/processor' }
```

Dica de Mestre: Entendendo as Ferramentas de Desenvolvimento

`cds-tsx` **(Recomendado)** 

Utiliza o motor ``tsx``. Extremamente rápido, pois foca apenas na transpilação, sem checagem de tipos. Ideal para o ciclo de desenvolvimento ágil.

`cds-ts` **(Alternativa)**

Utiliza o motor ``ts-node``. Um pouco mais lento, pois pode realizar checagem de tipos durante a execução.



Não use em Produção!

Ferramentas como ``cds watch``, ``cds-tsx`` e ``cds-ts`` são exclusivas para desenvolvimento. Para produção, sempre compile seu código para JavaScript para obter a melhor performance.

Fase 3: Garantindo a Qualidade com Testes

Como configuro e executo meus testes escritos em TypeScript?

1. Instale a dependência

```
Roboto Mono Regular  
npm install -D ts-jest
```

2. Configure o Jest

No seu `jest.config.js`, adicione o preset.

```
module.exports = {  
  preset: "ts-jest"  
};
```

3. Informe o Ambiente ao CDS

Crie `test/setup.ts` para que o CAP saiba que deve procurar por arquivos `*.ts`.

```
// ./test/setup.ts  
process.env.CDS_TYPESCRIPT = "true";
```

4. Execute os testes

```
jest
```

Fase 4: Preparando para Produção

Qual é o processo de build para um projeto TypeScript?

- **O Comando**

Use o `cds build` padrão. Uma tarefa de build dedicada, parte do `cds-typer`, gerencia a transpilação de TypeScript para JavaScript.

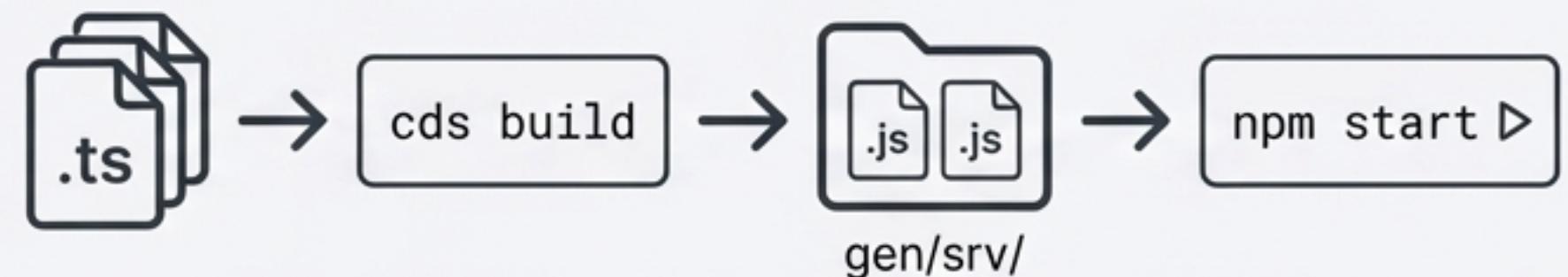
- **O Resultado**

O código JavaScript compilado é gerado na pasta `gen/srv/`. É este código que será implantado.

- **Testando o Build Localmente**

Para simular o ambiente de produção, execute a aplicação a partir da pasta gerada.

1. cds build
2. cd gen/srv
3. npm start



Fase 5: Otimizando a Experiência do Desenvolvedor

Como posso tirar o máximo proveito da tipagem nas APIs do CAP?

Apresentando o `@cap-js/cds-types`: Para obter IntelliSense e checagem de tipos nas APIs do CAP (Request, Service, etc.), adicione o pacote de tipos.

```
npm add @cap-js/cds-types
```

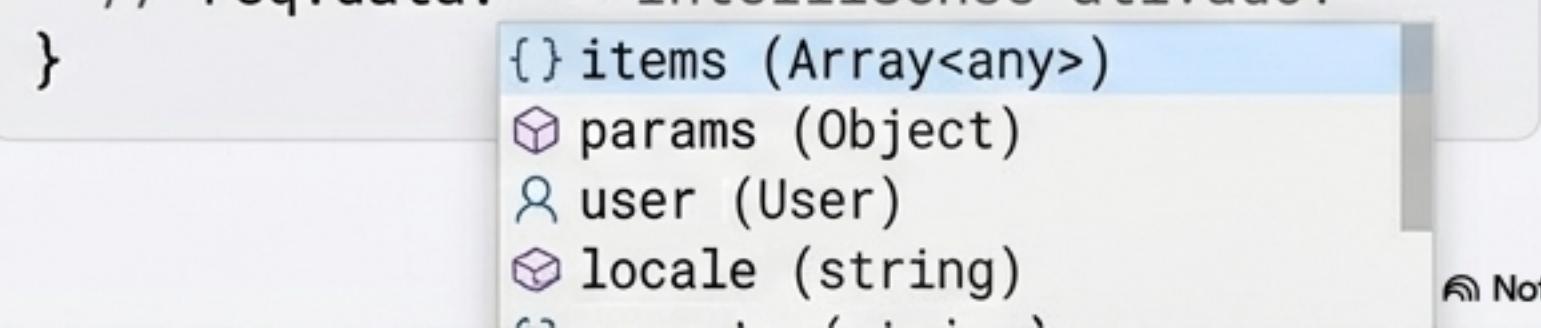
Como usar: Importe os tipos diretamente de `@sap/cds`.

Antes (JavaScript ou sem tipos)

```
function myHandler(req) {  
    // req.data.? -> Sem ajuda do editor  
}
```

Depois (TypeScript)

```
import { Request } from '@sap/cds'  
  
function myHandler(req: Request) {  
    // req.data. -> IntelliSense ativado!  
}
```



A screenshot of a TypeScript code editor showing an Intellisense dropdown menu. The menu lists several properties of the 'req.data' object: 'items (Array<any>)', 'params (Object)', 'user (User)', and 'locale (string)'. The first item, 'items', is highlighted with a blue background.

- {} items (Array<any>)
- 📦 params (Object)
- 👤 user (User)
- 📦 locale (string)

Boas Práticas de Importação de Tipos

✓ FAÇA

Importe apenas da fachada `@sap/cds`.

```
import { Request } from '@sap/cds'
```

✗ NÃO FAÇA

Nunca importe de caminhos internos da API.

```
import { Request } from  
'@sap/cds/apis/events'
```



Tipos Mantidos pela Comunidade

Como `@sap/cds` é uma biblioteca JavaScript, as tipagens podem não estar sempre 100% atualizadas com a última release. Sinta-se à vontade para contribuir no GitHub!



E quanto aos tipos das minhas próprias entidades do CDS?

- **O Poder do `cds-typer`:** O pacote [cds-typer](#), adicionado na configuração inicial, deriva definições TypeScript diretamente do seu modelo de dados CDS.
- **O Benefício:** Isso fornece autocompletar e segurança de tipo para suas entidades (ex: `Books`, `Authors`) dentro dos seus handlers de serviço, prevenindo erros e acelerando o desenvolvimento.

```
import { Book } from '#cds-models/sap/capire/bookshop';

this.before('CREATE', Book, req => {
    // req.data.|
```

})

T title: string
ID: number
T author: Author
stock: number
⌚ createdAt: Date

Resumo da Jornada TypeScript no CAP



Setup

```
cds add typescript
```



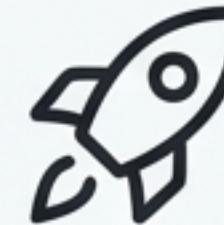
Desenvolvimento

```
cds watch
```



Testes

```
ts-jest
```



Build

```
cds build
```



Produtividade

```
@cap-js/cds-types  
&  
cds-typer
```

Próximos Passos e Recursos

Explore um Projeto Completo



Confira a aplicação de exemplo [SFlight](#), que apresenta handlers e UI em TypeScript.

Consulte a Documentação



Aprofunde seus conhecimentos na documentação oficial do TypeScript no CAP e do [cds-typer](#).

Junte-se à Comunidade



Contribua com as tipagens e ajude a aprimorar o ecossistema. Abra um Pull Request ou uma Issue no GitHub.

Obrigado!

Perguntas?