# Guia de Estrutura Backend Express com TypeScript

## Introdução

Este documento apresenta uma estrutura modular recomendada para um backend construído com Express e TypeScript.  
A organização é inspirada em padrões de projetos escaláveis e fáceis de manter, similar à estrutura de apps no Django.

## Estrutura de Pastas

/project-root  
|-- /src  
|-- server.ts # Ponto de entrada do app, configura express e rotas globais  
|-- /config # Configurações gerais (dotenv, DB, variáveis de ambiente)  
|-- /middlewares # Middlewares (autenticação, logger, validação, etc)  
|-- /modules # Módulos de negócio, tipo apps no Django  
|-- /users  
|-- users.routes.ts # Define rotas (URL -> controller)  
|-- users.controller.ts # Recebe req/res, chama service, manda resposta  
|-- users.service.ts # Lógica de negócio, consulta DB, regras  
|-- users.model.ts # Definição do schema/modelo (ex: mongoose)  
|-- /auth  
|-- auth.routes.ts  
|-- auth.controller.ts  
|-- auth.service.ts  
|-- auth.model.ts  
|-- /utils # Helpers genéricos, funções utilitárias  
|-- package.json  
|-- tsconfig.json  
|-- .env

## Descrição das Pastas e Arquivos

* **server.ts**: Arquivo principal que inicia o servidor Express, configura middlewares globais e conecta as rotas.
* **/config**: Contém configurações gerais da aplicação, como variáveis de ambiente (dotenv) e configuração de banco de dados.
* **/middlewares**: Middlewares customizados que atuam no ciclo da requisição, como autenticação, logging e validação.
* **/modules**: Agrupa módulos ou funcionalidades específicas da aplicação, seguindo uma estrutura modular para melhor organização e escalabilidade.
* **/modules/users**: Exemplo de módulo que contém tudo relacionado a usuários: rotas, controllers, serviços e modelos.
* **users.routes.ts**: Define as rotas específicas do módulo users, associando URLs aos controllers.
* **users.controller.ts**: Controlador que recebe as requisições, processa entrada e saída, e chama os serviços para lógica de negócio.
* **users.service.ts**: Contém a lógica de negócio do módulo users, incluindo acesso ao banco de dados e regras específicas.
* **users.model.ts**: Define o modelo ou schema do usuário, por exemplo com Mongoose para MongoDB.
* **/utils**: Funções auxiliares genéricas que podem ser usadas em qualquer parte da aplicação para evitar duplicação de código.
* **package.json**: Arquivo de configuração do NPM que gerencia dependências e scripts de execução.
* **tsconfig.json**: Configuração do compilador TypeScript definindo regras de compilação e diretórios.
* **.env**: Arquivo de variáveis de ambiente, geralmente não versionado, que guarda informações sensíveis e configurações específicas do ambiente.