Estrutura do Projeto

Iremos criar quatros pastas config,modules,shared,service.

config - configurações de bibliotecas externas, como por exemplo, autenticação, upload, email, etc.

modules - abrangem as áreas de conhecimento da aplicação, diretamente relacionados com as regras de negócios. A princípio criaremos os seguintes módulos na aplicação: customers, products, orders e users.

shared - módulos de uso geral compartilhados com mais de um módulo da aplicação, como por exemplo, o arquivo server.ts, o arquivo principal de rotas, conexão com banco de dados, etc.

services - estarão dentro de cada módulo da aplicação e serão responsáveis por todas as regras que a aplicação precisa atender, como por exemplo:

* A senha deve ser armazenada com criptografia;
* Não pode haver mais de um produto com o mesmo nome;
* Não pode haver um mesmo email sendo usado por mais de um usuário;
* E muitas outras...

# Configurando as importações

Podemos usar um recurso que facilitará nossos imports dentro da aplicação.

Intalação do tsconfig-paths => yarn add -D tsconfig-paths

Iniciamos configurando o objeto paths do tsconfig.json, que permite criar uma base para cada path a ser buscado no projeto, funcionando de forma similar a um atalho, ex:

"paths": {

"@config/\*": ["src/config/\*"],

"@modules/\*": ["src/modules/\*"],

"@shared/\*": ["src/shared/\*"]

}

Se formo usar essa estrutura de path temos que passar o parâmetro:

-r tsconfit-path/register

Ficaria algo assim: "ts-node-dev -r tsconfig-paths/register --inspect --transpile-only --ignore-watch node\_modules src/shared/http/server.ts"

temos que descomentar o baseUrl: “./” do ts.config.json

dentro de shared criamos pasta routes e criamos o arquivo para rotas index.ts

import { Router } from "express";

const routes = Router();

routes.get('/', (req, res) => {

    res.send('ok')

})

export default routes;

depois de colocarmos o código das rotas , importamos o routes no server.ts e usamos ele no app.use(routes)