Perguntas Teóricas

Pergunta 1.1

Descreva como você organizaria um projeto usando NestJS em uma arquitetura de microserviços. Quais seriam os principais componentes e como eles interagiriam?

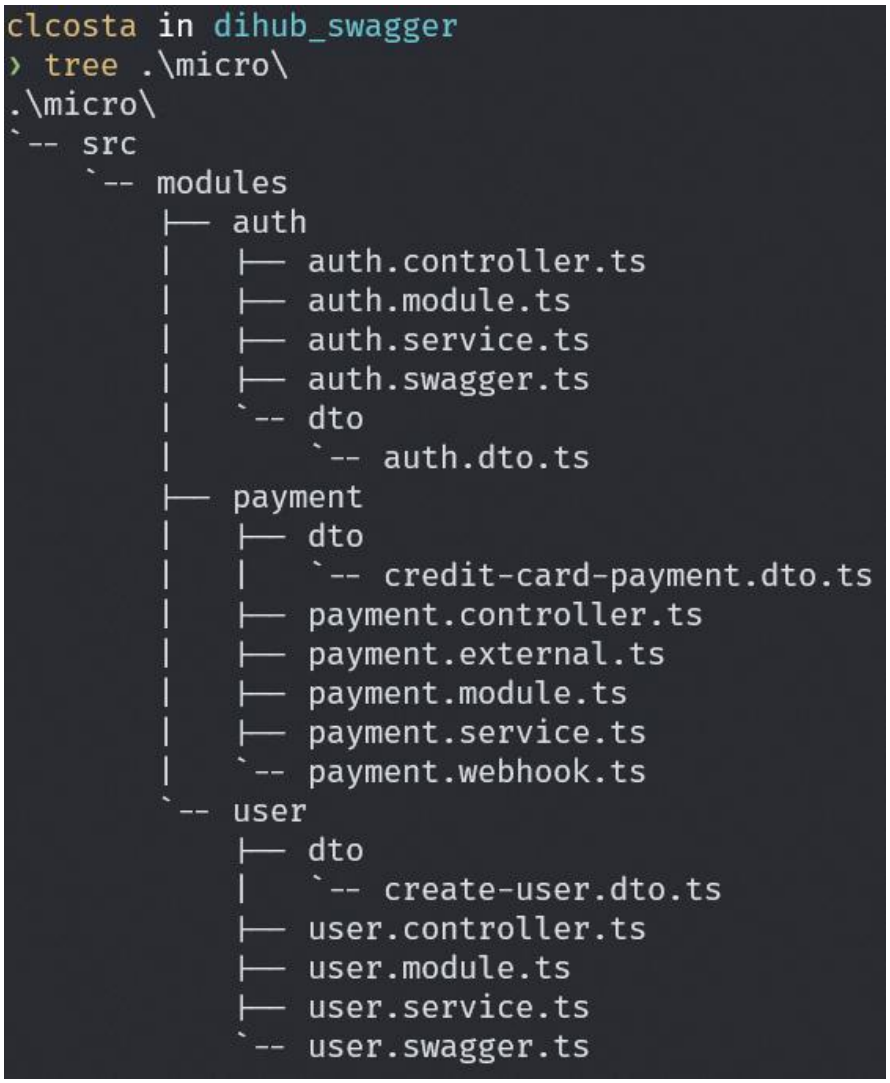
Para organizar projetos em Nest eu utilizaria a arquitetura recomendada pelo próprio framework, uma arquitetura modularizada. Microserviços funcionam como uma pequena aplicação independente, o que se encaixa perfeitamente na arquitetura modular.

Cada modulo teria sua própria estrutura para rodar independente, seguindo a responsabilidade única e apontando apenas para o seu próprio domínio.

Os principais componentes:

- **Controllers**: Responsáveis por lidar com a camada de requisição entre o cliente a aplicação. Não implementa a regra de negócio.
- **Services**: Responsável por implementar a regra de negócio e lógica concreta da aplicação.
- **DTO**: Validação de dados e facilita a transferência dos dados entre os componentes da aplicação.
- **Modulo**: Agrupa toda funcionalidade (controllers, services, adpaters, groups, etc.) isolando o componente, permitindo ser reutilizado de forma simples.
- **Mensageria:** Usado para comunicação interna entre microserviços, integração geralmente é realizada com RabbitMQ, Kafka ou até mesmo com Redis. Mensageria é recomendado para sistemas distribuídos em que a comunicação será assíncrona.

Exemplo de estruturas de pastas:



Legenda: Um exemplo de estruturas de pastas mostrada no terminal. Cada linha representa um ramo de pastas. Na imagem cada pasta dentro de *modules* poderias ser um microserviço operando de forma independente dos outros serviços.

Como você abordaria o gerenciamento de estado e a comunicação entre serviços em uma aplicação distribuída?

Para gerenciar os estados de uma aplicação distribuída é comum utilizar um armazenamento persistente, ou seja, um banco de dados. Porém, em casos em que é necessário armazenar o estado por um pequeno período ou que exige uma comunicação de IO (Input/Output) mais rápida pode se usar também cache, com bancos em memória ou até mesmo armazenamentos locais temporários.

A comunicação no geral bem simples, pode se utilizar comunicação via API's REST, Graphql ou uma comunicação assíncrona por mensageria. O simples geralmente já resolve, a primeira abordagem seria API REST.

Pergunta 1.2

Qual a diferença entre tipos e interfaces em TypeScript? Quando você usaria um ou outro?

A diferença é que **interfaces** são utilizadas para definir a estrutura de objetos e classes, definindo a implementação desejada, sendo mais específicas para contratos. Já os **tipos** são *alias* para tipos primitivos, sendo assim, eles são mais flexíveis podendo se entender para funções e uniões de tipos.

Ambos não é *transpilados* para JavaScript no build, existindo apenas na verificação de tipos do *compile*.

Explique o conceito de generics em TypeScript e forneça um exemplo prático.

Tipos genéricos são utilizados quando temos um código que é reutilizado para diferentes tipos, passando apenas o tipo como parâmetro evitando assim de ter que escrever um código para cada tipo.

```ts
type User = {
    id: number;
}
type Post = {
    id: number;
}

function showId<T extends User | Post>(instance: T) {
    console.log(instance.id);
}

const user: User = { id: 1 };
const post: Post = { id: 2 };

showId(user); // Checagem de tipo infere o tipo da variável user como User
showId(post); // Checagem de tipo infere o tipo da variável post como Post
showId<User>({ id: 1 }); // Digo explicitamente que o tipo é User para o objeto passado
showId('string'); // Erro de tipagem

```

Legenda: Um código de exemplo sobre como pode ser utilizado genéricos. No exemplo em questão o código define dois tipos contendo o atributo ***id***, cria uma função que aceita tanto o tipo ***User*** quanto o tipo ***Post*** como parâmetro e mostra o atributo ***id.***

Pergunta 1.3

Quais são as principais aplicações do Puppeteer e como ele pode ser utilizado para realizar web scraping?

As principais aplicações do puppeter são:

- Web Scraping
- Testing
- Automações de tarefas

Para fazer web scraping com Puppeteer, é importante entender como o framework funciona e como o site alvo é estruturado. O processo segue um paralelo com o passo a passo realizado pelo humano, podendo refazer todas as ações de um usuário comum e muito mais. Tendo em vista que, o puppeteer interage diretamente com o conteúdo da página, logo se necessário extrair alguma informação do site é possível obtê-la diretamente pelo HTML da página.