**Criando sua primeira aplicação Node.js com NestJS**

NestJS é um dos mais completos frameworks para NodeJs. É baseado no express e escrito em Typescript.

Este artigo irá mostrar como iniciar uma aplicação do zero em menos de 1 hora. Ao longo deste artigo iremos criar uma aplicação simples para cadastro de usuários. Porém além de cadastrar usuários a aplicação também fará um tracing da requisição.

Pré-requisitos

1 - NestJs cli: para gerar um template do código precisamos instalar a cli, para isso usamos o comando “npm i -g @nest/cli

2 – Uma base de dados, neste exemplo usaremos postgres

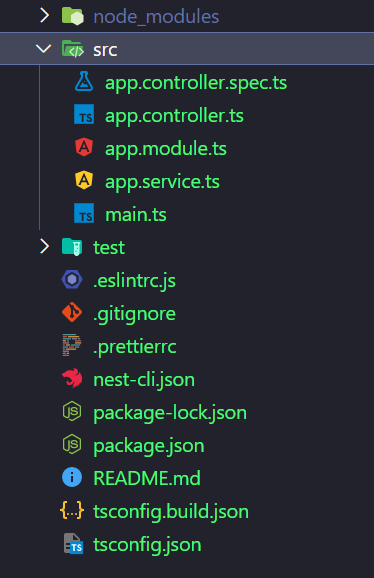
3 – Ter o docker instalado

4 – Ter a imagem do Jaeger instalada

Start de uma nova aplicação

Para criarmos uma nova aplicação basta executar o comando “nest new nome\_projeto”. E depois escolher o gerenciador de pacotes da sua preferência. Logo depois para checarmos se está tudo ok basta executar o comando “npm start:dev” e a aplicação irá iniciar o servidor na porta 3000.

A aplicação possui a seguinte configuração (figura abaixo), com uma controller, service e app module pré-configurados.



Criando User Controller e Module

Antes de criar sua primeira rota, é necessário criar um módulo. Módulos em aplicações Nest organizam a estrutura da aplicação. Para criar o módulo de usuários é necessário rodar o comando “nest generate module users”. Este comando irá criar uma pasta chamada users, com um arquivo chamado users.module. O arquivo contendo o módulo principal “app.module” sera atualizado com a importação do users.module.

Agora precisamos de uma controller, para isso basta rodar o comando “nest generate controller users”. Este comando irá criar a controller de usuários em conjunto com um arquivo de testes dentro da pasta users.

Usando rota para salvar objetos e devolve-los

Para sermos capazes de criar um usuário através da rota recém criada, precisamos de uma base de dados para armazena-los.

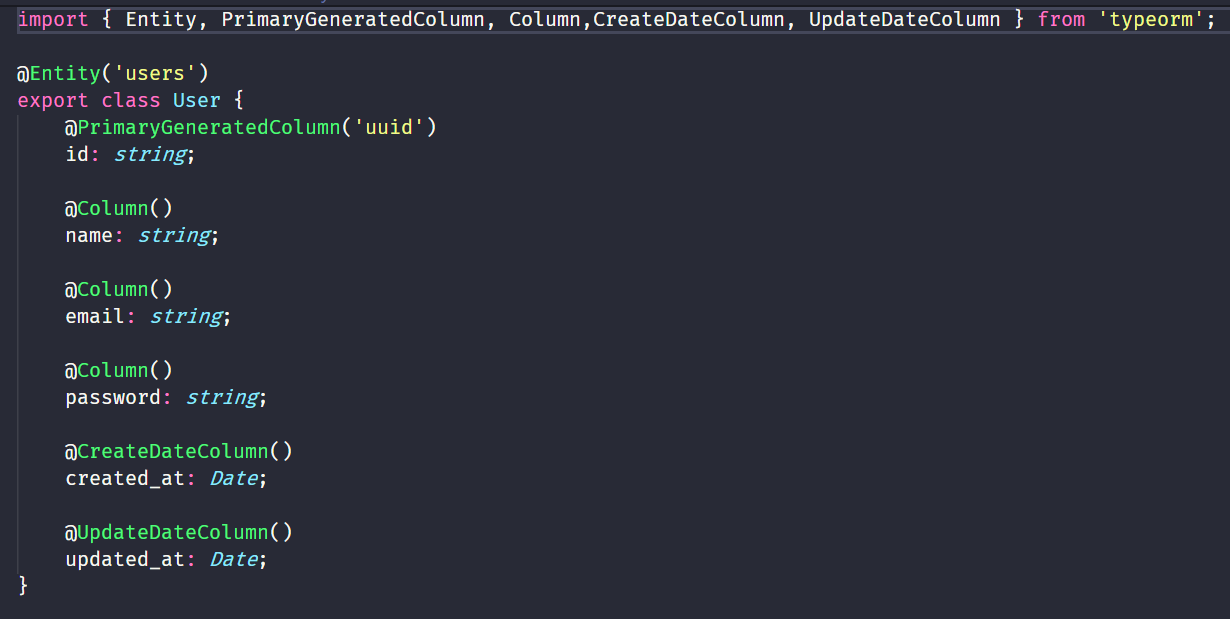
Para este exemplo, usaremos postgres e para manipular a base de dados usaremos o typeorm que também é escrito em typescript e é bastante recomendado por desenvolvedores NestJs .

Adicionaremos typeorm e pg junto as dependências pelo comando

“npm install @nest/typeorm typeorm pg”.

Criando a entidade de usuários

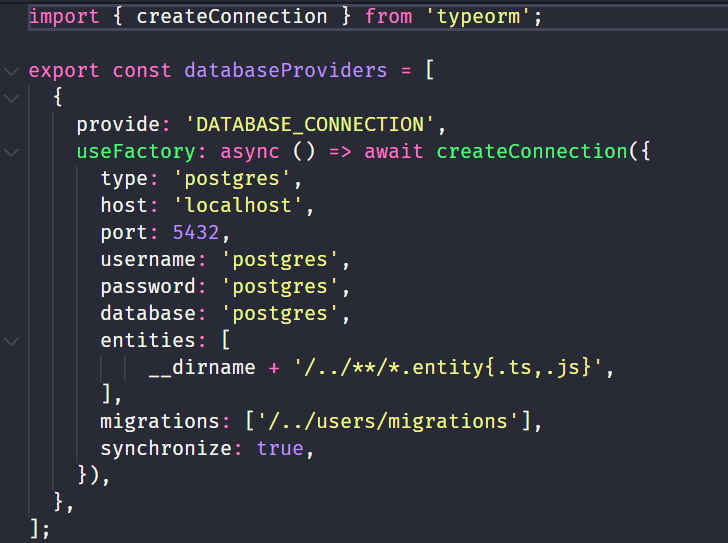
Agora vamos criar a entidade users, essa entidade irá demostrar como ficará salvo nosso usuário no banco de dados. Vamos criar de acordo com o padrão já existente em “users/users.entity.ts”. A figura abaixo demonstra como a entidade pode ser criada.



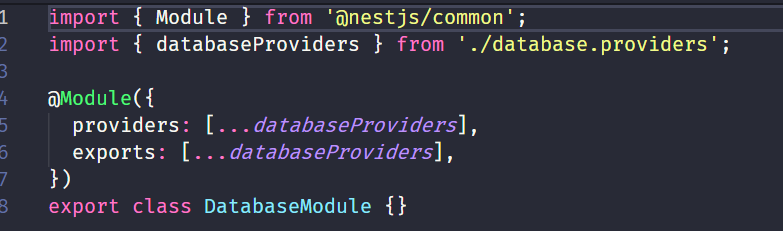
Neste exemplo iremos criar uma tabela com 6 campos, onde o campo id será nossa chame primária com auto incremento a cada usuário criado. Os campos created\_at e updated\_at representam quando foi criado um recurso e a última vez que foi atualizado.

Conectando aplicação ao banco de dados

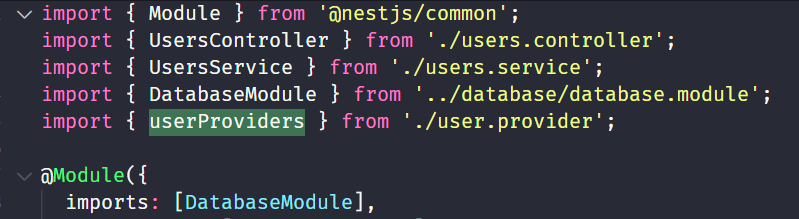
O primeiro passo que precisamos fazer é estabelecer a conexão com nosso banco de dados, criaremos um arquivo chamado database.providers e usando a função createConnection () importada do pacote typeorm. A função createConnection () retorna uma promessa e, portanto, temos que criar um provedor assíncrono.



Em seguida, precisamos exportar esses provedores para torná-los acessíveis para o restante do aplicativo. Para isso criaremos um módulo para organizar melhor o código, o chamaremos de database.module.

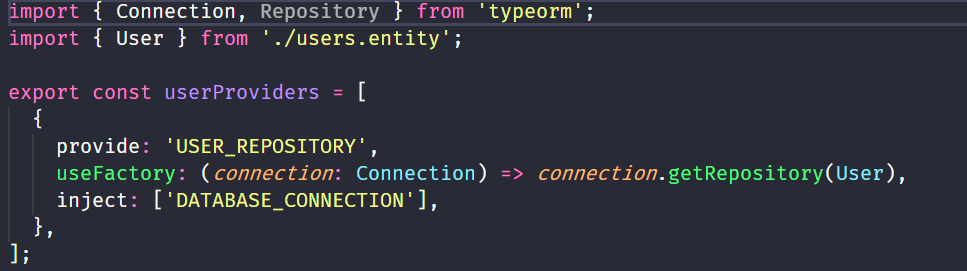


Logo depois precisamos importar o recém criado módulo e cadastrá-lo no módulo de user

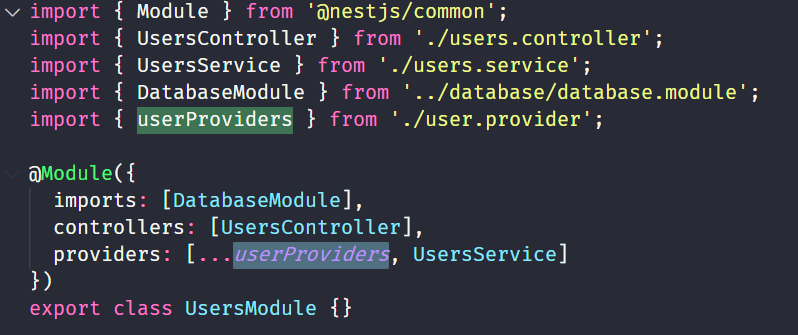


Criação do repositório de usuários

O TypeORM suporta o padrão de design do repositório, portanto, cada entidade tem seu próprio Repositório. Esses repositórios podem ser obtidos na conexão com o banco de dados. Para criar um repositório iremos criar um repositório provider, como se trata de um repositório de usuários, iremos criar um arquivo dentro do folder user chamado user.provider.



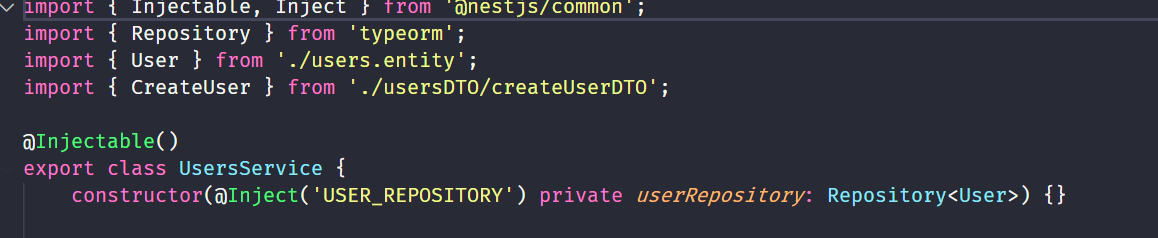
Após a criação é necessário cadastrar o provider no módulo de users.



Criação de um service para manipular consultas ao banco de dados

Agora que temos total conectividade ao banco de dados, é necessário a crição de uma classe que será ponte de comunicação da controller com o banco de dados. Neste caso iremos criar uma classe de serviço. Para criar um service no NestJs basta digitar o comando “nest generate service users”.

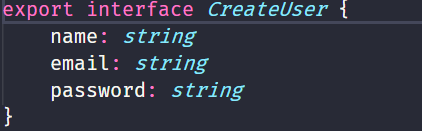
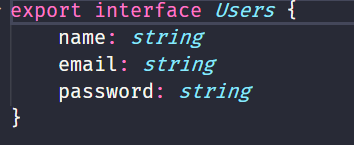
Após criado iremos injetar o nosso repositor recém criado como dependência de nosso service.



Feito iremos criar as funções básicas para criação, listagem, alteração e deleção de usuários.



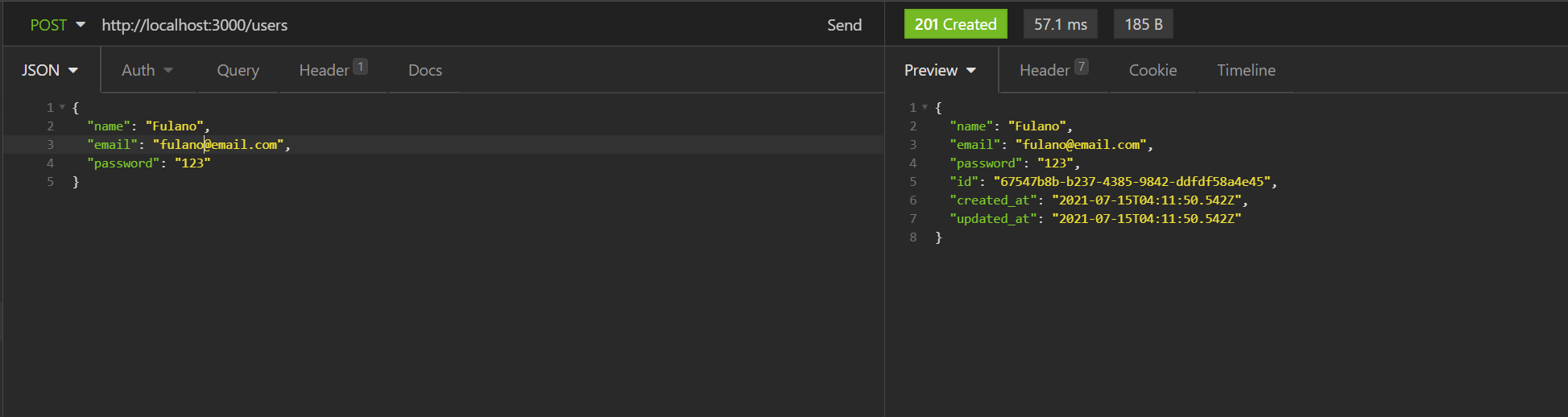
Obs: Para a representação da criação, listagem ou mesmo criação de um recurso foi criado dois DTOs. Estes DTOs são usados no service e na controller.

**

O último passo é importar nosso service em nossa controller, chamando as respectivas funções e passando os parâmetros corretamente.

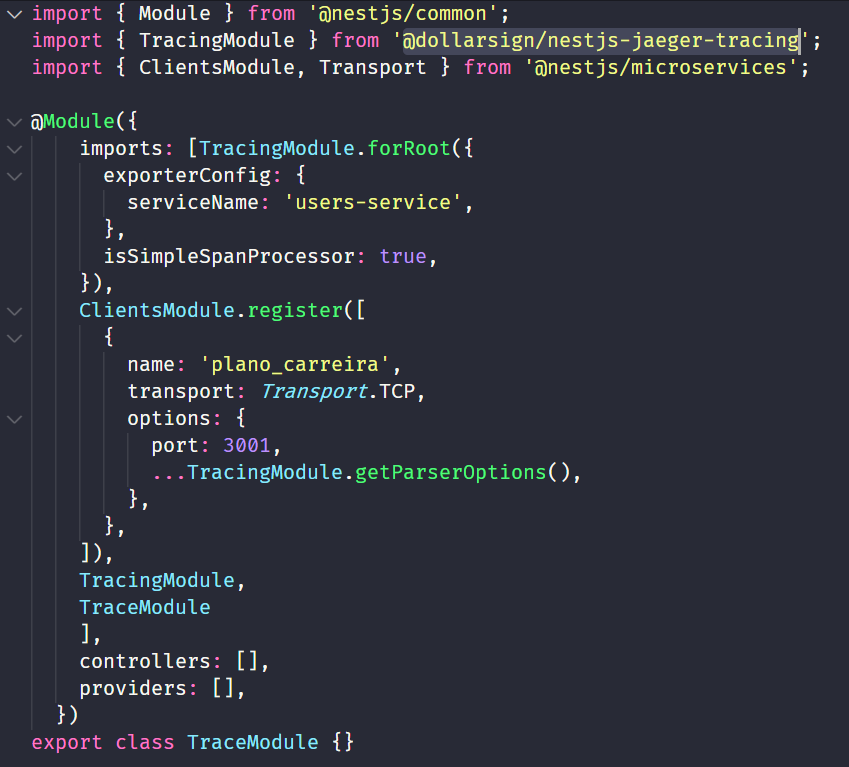


Agora que está tudo conectado é possível manipular o cadastro de usuários.



Configurando trancing no projeto

Para configurar o tracing da nossa aplicação precisamos instalar a seguinte dependência “npm install @dollarsign/nest-jaeger-tracing”. Após a instalação será necessária a criação de um módulo que irá receber as configurações do tracing.



Feito já é possível realizar o tracing dos requests feitos em nossa aplicação.

