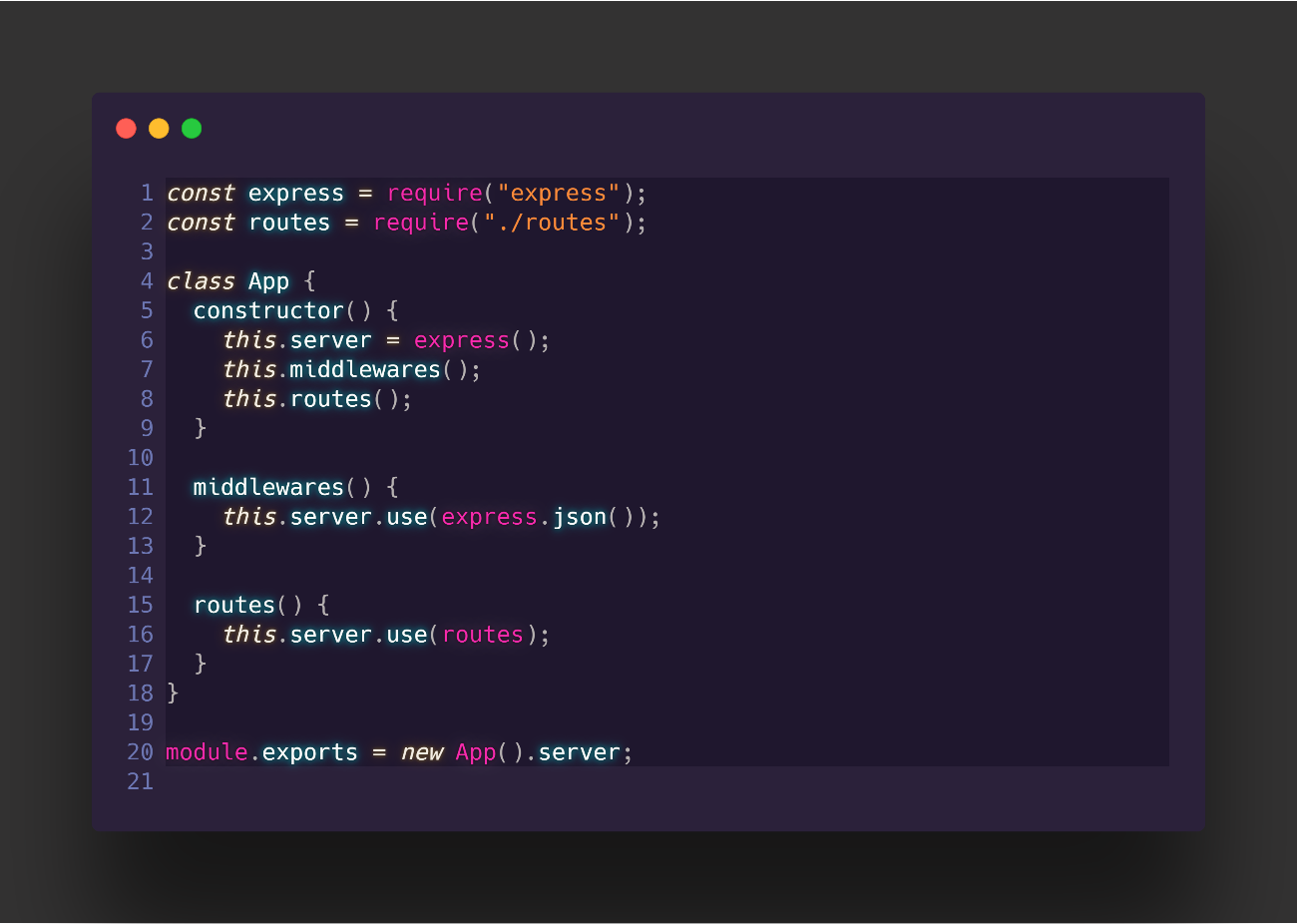
1. **Configurando a estrutura do Projeto**
   1. Crie uma, sugiro que tenha o nome **backendgeek**, para criar via cmd ou terminal use: **mkdir backendgeek**
   2. yarn init -y (para inicializar o projeto)
   3. yarn add express (instalar o pacote do express)
   4. criar pasta **src**
   5. dentro criaremos alguns arquivos:
      1. app.js
      2. server.js
      3. routes.js
   6. no arquivo **app.js,** configuremos a seguinte maneira:

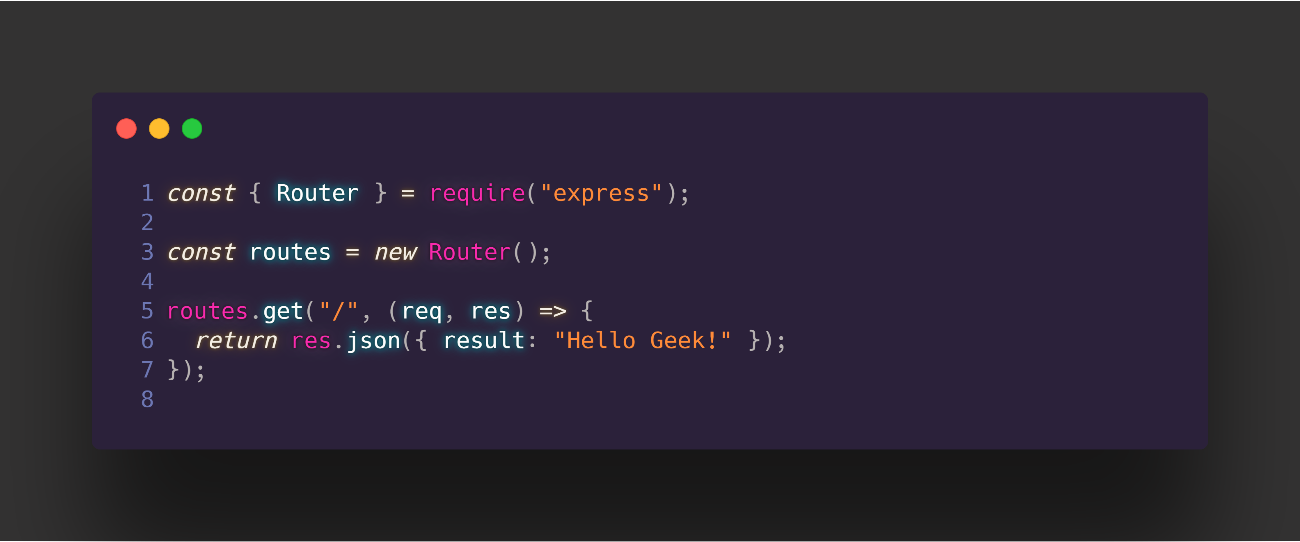


Note que importamos o nosso arquivo de rotas.

* 1. Agora nós iremos trabalhar dentro do nosso **server.js**

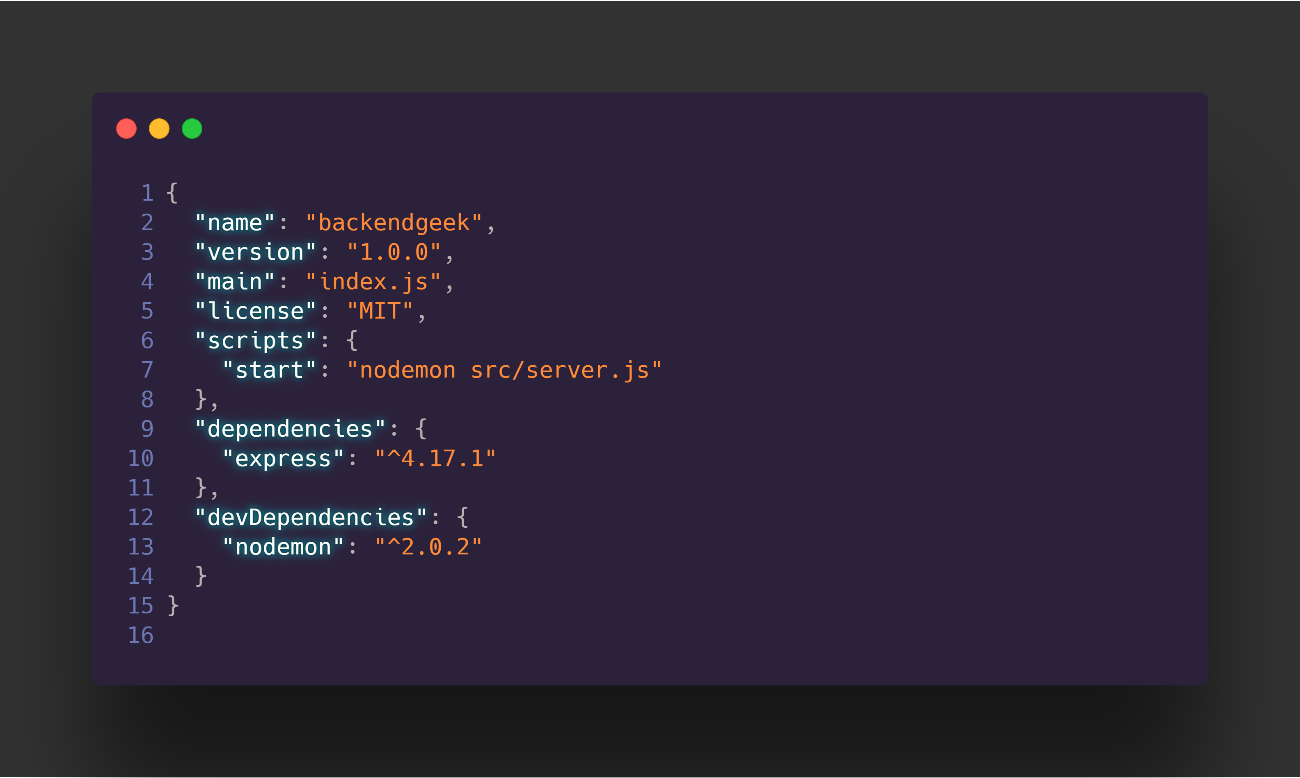


* 1. O nosso arquivo **routes.js** é onde ficará as nossas rotas, você pode cria-lo da seguinte maneira (note que já temos uma rota para testar):



* 1. Agora rodamos no terminal node src/server.js
  2. Basta acessar o seu navegador e digitar localhost:3333 que o result será exibido.

1. **Nessa aula nós iremos adicionar o NODEMON para rodar o nosso servidor.**
   1. yarn add nodemon -D
   2. agora vá até o Package.json e insira os **scripts**, o seu arquivo ficará assim:



* 1. agora você já pode rodar no terminal o comando **yarn start** assim o seu servidor ficará online.
  2. Para utilizarmos import e export default nós iremos instalar o sucrase
     1. Yarn add sucrase -D
     2. Na raiz do projeto criamos um arquivo chamado nodemon.json, que ficará assim:



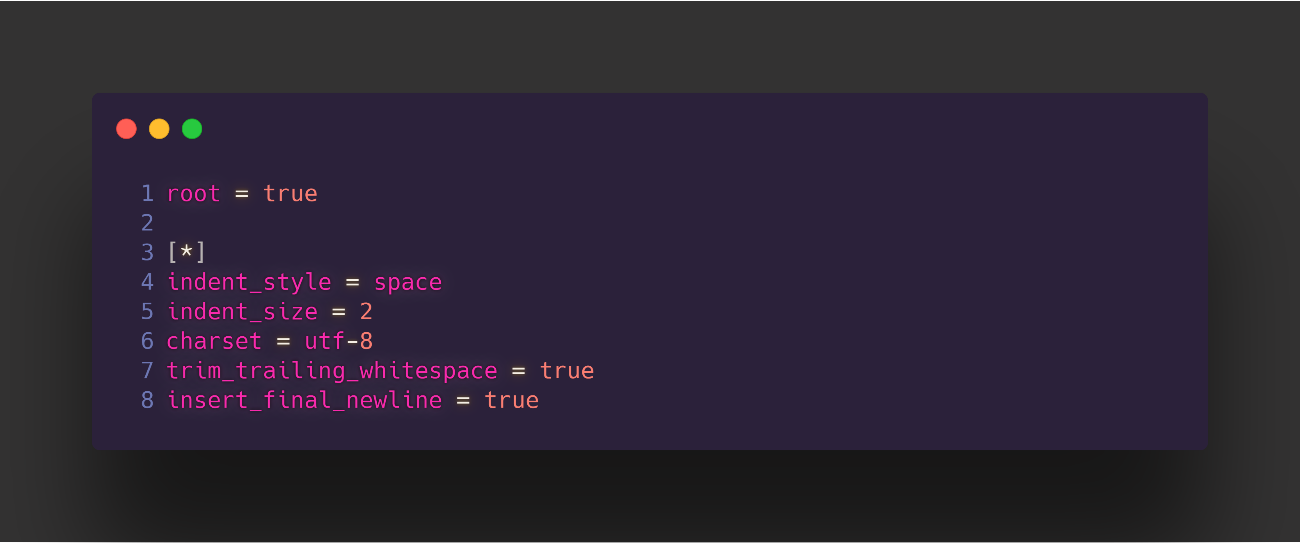
1. **Nessa aula nós iremos configurar o Eslint, Prettier e Editorconfig**
   1. yarn add eslint -D
   2. yarn eslint --init
      1. To check syntax, find problems, and enforce code style
      2. JavaScript modules (import/export)
      3. None of these
      4. Does your project use TypeScript? (y/N) colocamos **N**
      5. ( ) Browser - (\*) Node (selecionamos o node com a barra de espaço)
      6. Use a popular style guide
      7. Airbnb: <https://github.com/airbnb/javascript>
      8. JavaScript
      9. Would you like to install them now with npm? (Y/n) **Y**
      10. agora deletamos o arquivo Package-lock.json e rodamos o comando **yarn**
   3. agora vamos configurar o **prettier** para trabalhar junto do **eslint**
      1. yarn add eslint-config-prettier
      2. yarn add eslint-plugin-prettier
      3. vamos até o arquivo .eslintrc.js e o deixamos da seguinte maneira:



* + 1. agora criamos um arquivo chamado **.prettierrc** na raiz do nosso projeto, e o deixamos com as seguintes configurações:



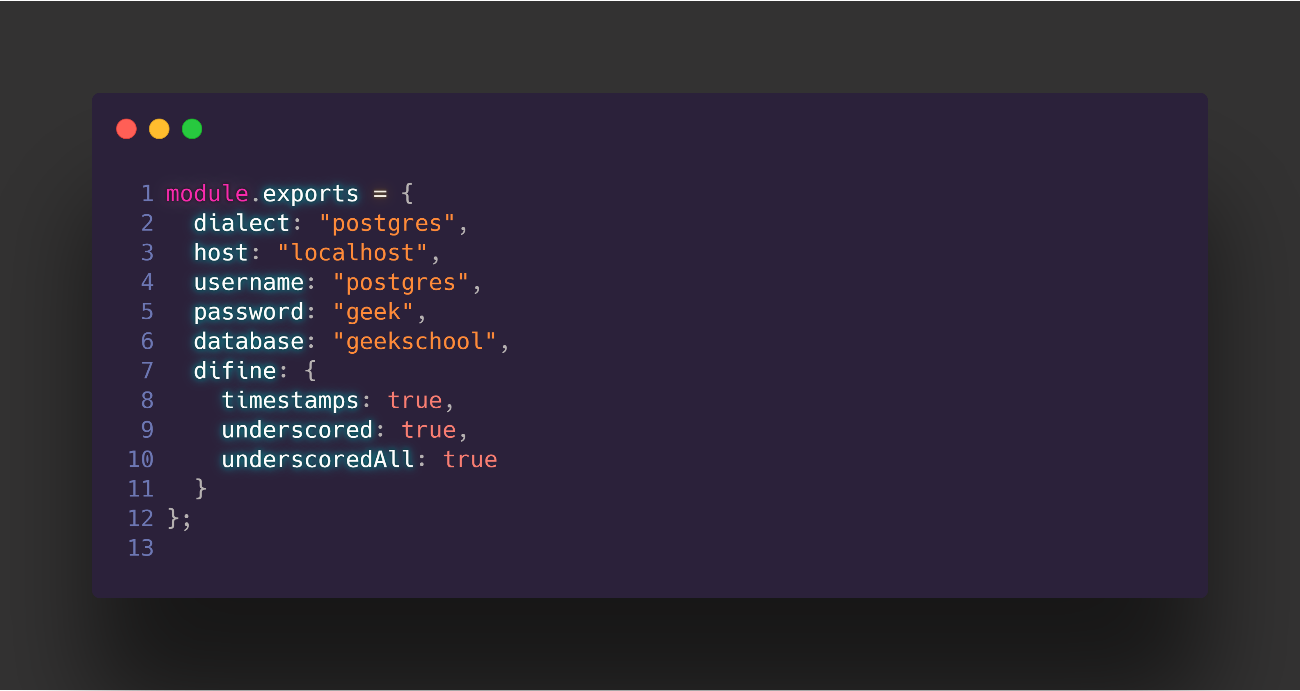
* 1. Agora iremos configurar o editorconfig (é o mais fácil)
     1. Clicamos com o botão direito na raiz do projeto e depois em: Generate editorconfig
     2. Abrimos o arquivo gerado na raiz (.editorconfig) e deixamos da seguinte maneira:



1. **Nessa aula nós iremos trabalhar com Docker**
   1. Criamos o nosso container para o banco de dados mysql
      1. **docker run --name** database **-e POSTGRES\_PASSWORD=**docker -p 5432:5432 **-d postgres**  
         O que está em **azul** é comando do Docker e o que está em **vermelho** é parâmetro.
   2. Agora digite docker ps e veja se o nosso banco de dados **geekdb** está funcionando.
2. **Nessa aula nós iremos configurar o nosso ORM Sequelize**
   1. começamos instalando duas dependências:
      1. yarn add sequelize
      2. yarn add sequelize-cli -D
   2. agora iremos adicionar algumas pastas a estrutura do nosso projeto (pastas dentro de **src**):
      1. criamos uma pasta chamada: **configs**
         1. dentro colocamos um arquivo **db.js**
      2. criamos uma pasta **database**
         1. dentro criamos uma pasta **migrations**
      3. agora criamos uma pasta chamada **apps**
         1. dentro de **apps** criamos uma pasta **models** e outra **controllers**
   3. nesse ponto devemos criar um arquivo na raiz do projeto chamado: **.sequelizerc,** esse arquivo ficará da seguinte maneira:



1. yarn add pg (para usar o postgres)
2. yarn add pg-hstore (para usar o postgres)
3. dentro do nosso arquivo src/configs/db.js faremos a seguinte configuração:

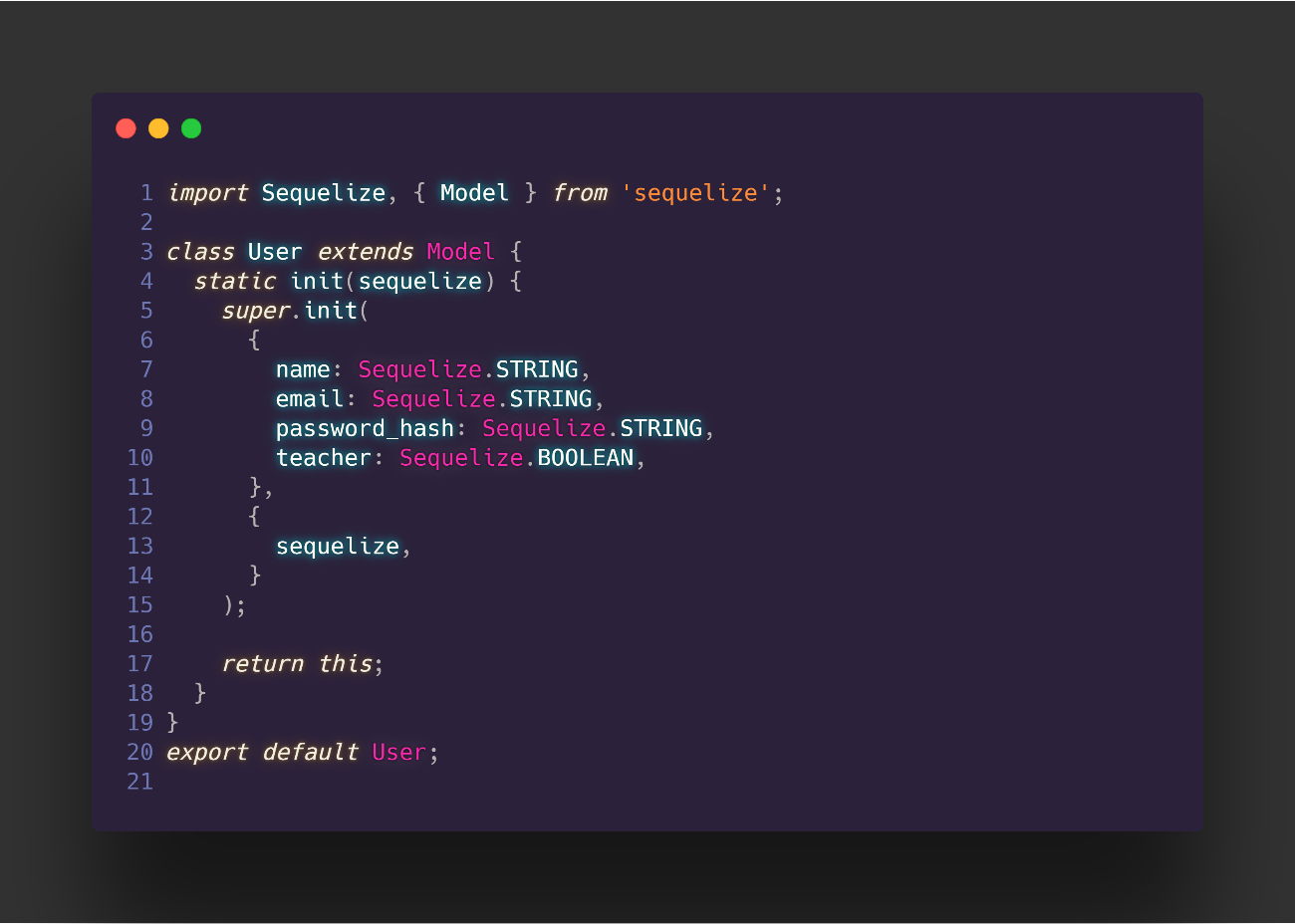


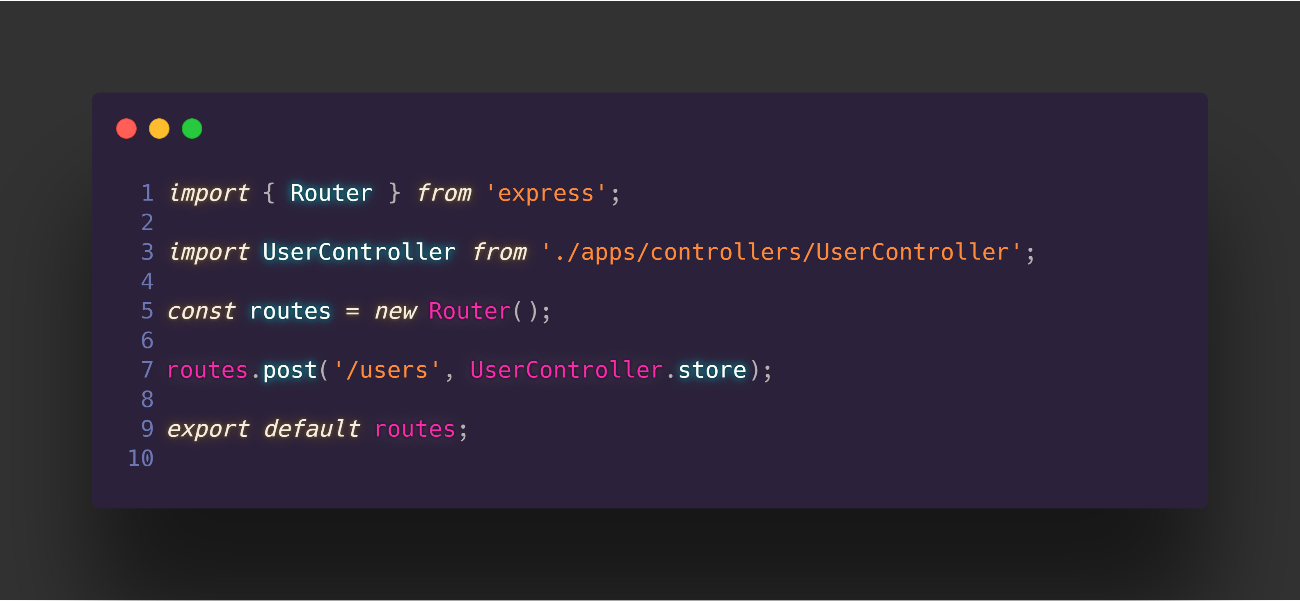
1. **Nessa aula nós iremos criar a nossa primeira migration (users)**
   1. Para criar utilizamos o seguinte comando:
      1. yarn sequelize migration:create --name=create-users
         1. agora dentro da pasta src/database/migrations nós teremos um novo arquivo com um nome cheio de números, esse arquivo ficará assim:

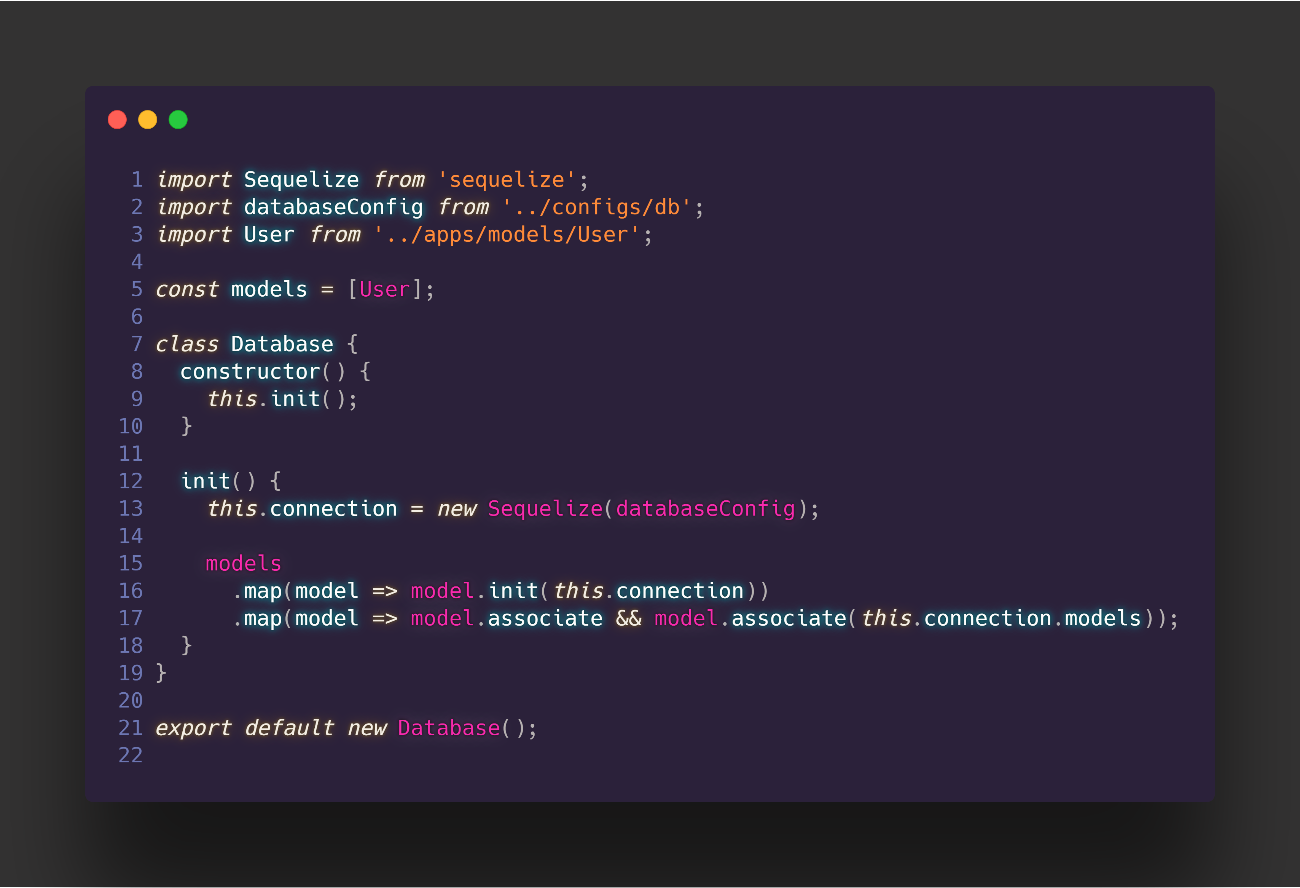


* 1. agora rodamos o seguinte comando: **yarn sequelize db:migrate**
     1. se tudo correr bem, não teremos nenhum erro no console.
     2. Agora basta abrir pgadmin4 e verificar se existe uma tabela **users** lá.

1. **Nessa aula nós iremos criar o nosso Model e Controller de users**
   1. Para isso vamos até a pasta src/apps/models e criamos um arquivo chamado **Users.js**, esse arquivo ficará assim:



* 1. Agora criaremos um arquivo UserController.js dentro de src/apps/controllers, nós já criaremos o nosso método store, que ficará assim: 
  2. Depois iremos em routes.js e criaremos a nossa primeira rota para inserir um usuário, que ficará assim:
  3. Agora iremos criar um arquivo que irá fazer a “conexão” entre o nosso banco de dados e o nosso model, esse arquivo ficará dentro de src/database e se chamará **index.js**, o seu conteúdo será o seguinte:



* 1. Agora dentro de routes.js importamos o arquivo index.js da pasta database, faremos isso da seguinte maneira:

import './database';

1. **Nessa aula nós iremos realizar o teste de inserção de usuário no banco de dados:**
   1. Para isso vamos até o postman e criaremos uma nova rota do tipo **POST**