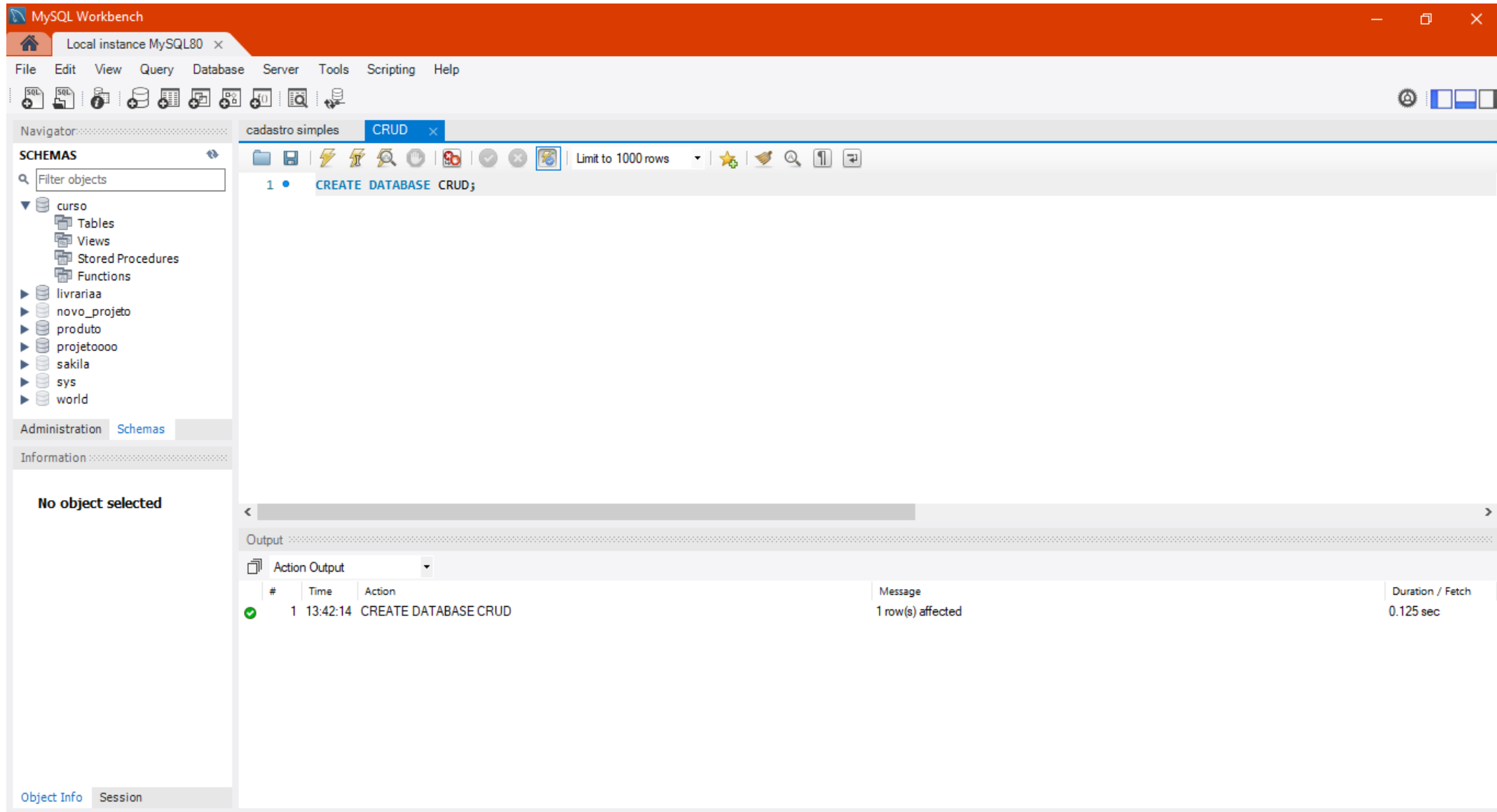


## Implementando e tratando uma conexão com o Banco de Dados através do Sequelize.

★ Inicialmente criei uma database no MySQL para receber os dados. Note que não há nenhuma tabela criada, apenas o banco de dados.

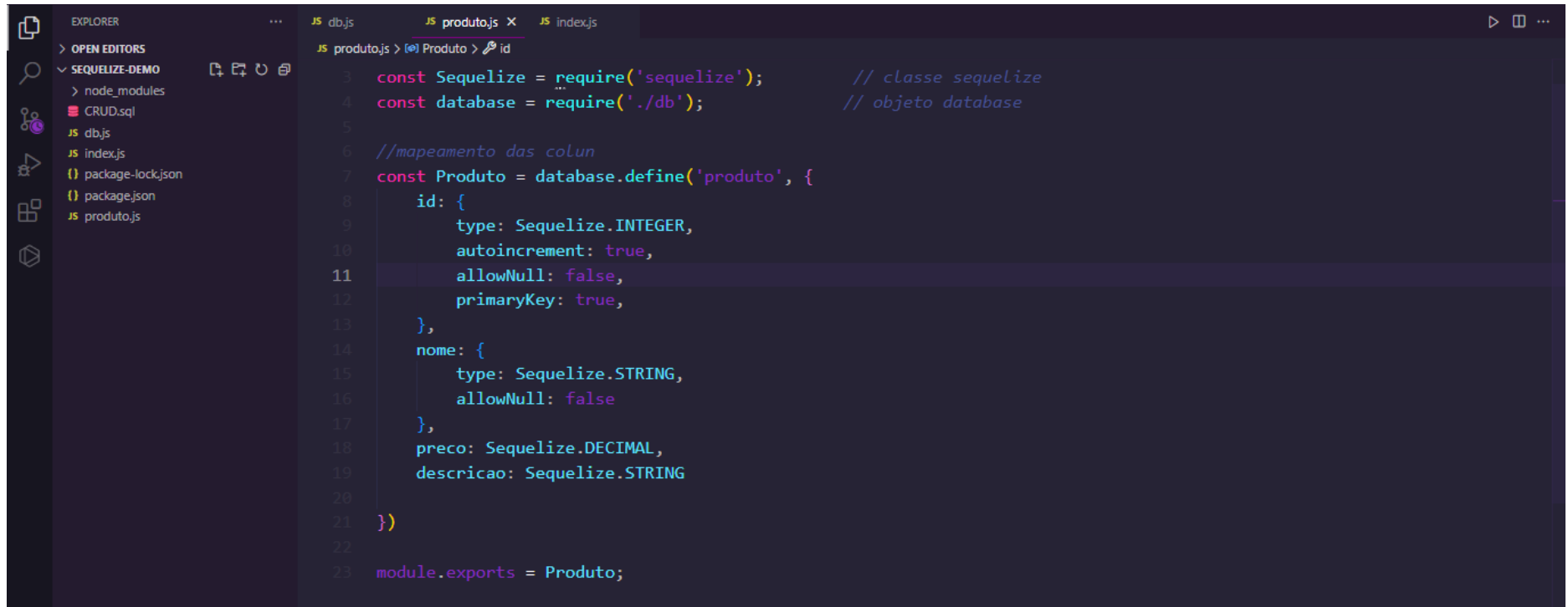


- ★ Iniciei um novo projeto Node.js executando ***npm init*** e criei o arquivo package.json.
- ★ Instalei o Sequelize e os *drivers* do banco de dados do MySQL com o comando no terminal ***npm install sequelize sequelize-cli mysql2***.
- ★ Após isso criei um arquivo ./db para configurar a conexão com o banco de dados. Criei uma classe Sequelize, instanciei a classe e adicionei os dados necessários.

A screenshot of the Visual Studio Code editor interface. The Explorer sidebar on the left shows a project structure with files like db.js, index.js, produto.js, package-lock.json, package.json, and produto.js. The main editor window displays the content of db.js, which contains the Sequelize configuration code. The code is as follows:

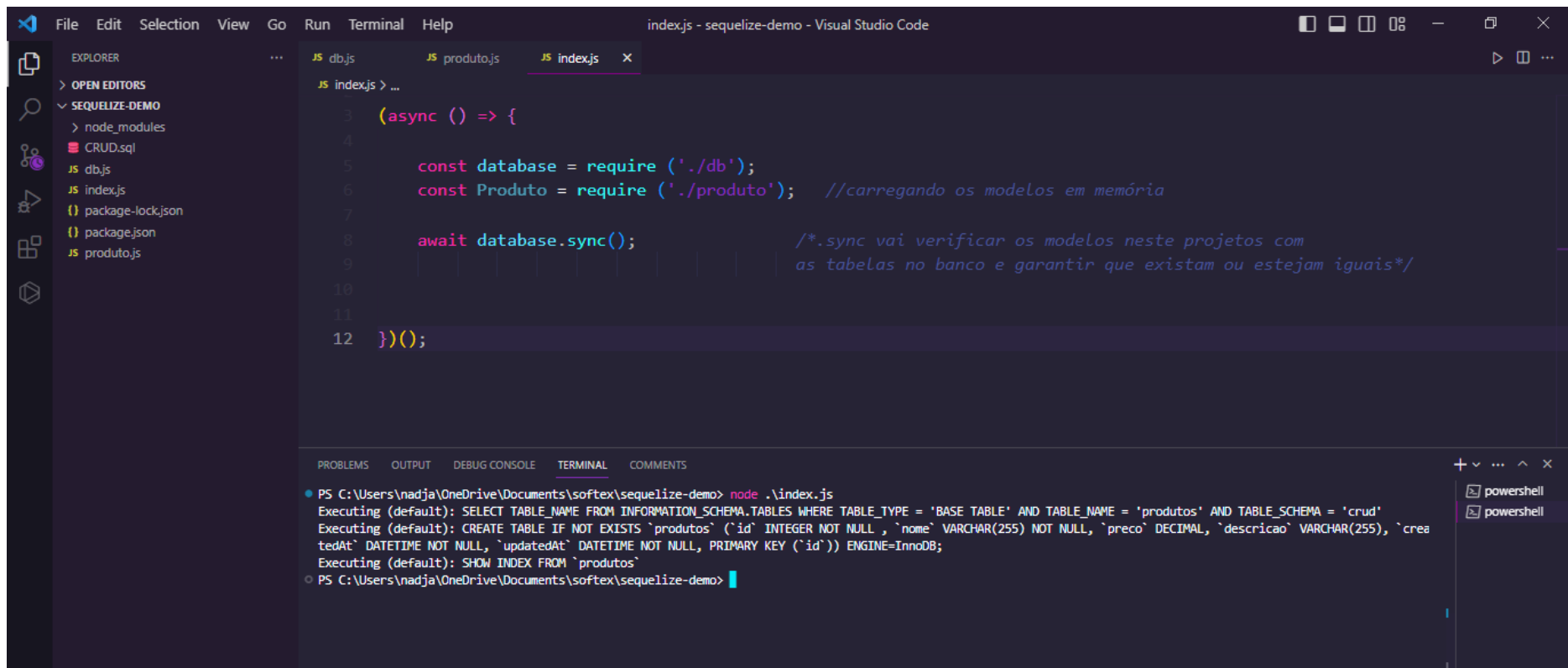
```
4 const Sequelize = require('sequelize');
5 const sequelize = new Sequelize ('crud', 'root', '1234',{
6     dialect: 'mysql',
7     host: 'localhost',
8     port: 3306
9 } )
10
11 module.exports = sequelize;
```

- ★ Criei um arquivo (./produto.js) e adicionei o modelo de dados. É o objeto que fará o mapeamento das colunas da tabela no banco de dados



```
1  const Sequelize = require('sequelize');           // classe sequelize
2
3  const database = require('./db');                 // objeto database
4
5
6  //mapeamento das colun
7  const Produto = database.define('produto', {
8    id: {
9      type: Sequelize.INTEGER,
10     autoincrement: true,
11     allowNull: false,
12     primaryKey: true,
13   },
14   nome: {
15     type: Sequelize.STRING,
16     allowNull: false
17   },
18   preco: Sequelize.DECIMAL,
19   descricao: Sequelize.STRING
20 },
21 {
22   //options
23 })
24
25 module.exports = Produto;
```

★ No arquivo `./index` eu vou utilizar o modelo criado antes. Ao executar a tabela irá ser criada.



The screenshot shows the Visual Studio Code interface with the file `index.js` open in the editor. The file contains the following code:

```
1  JS index.js > ...
2
3  (async () => {
4
5      const database = require ('./db');
6      const Produto = require ('./produto'); //carregando os modelos em memória
7
8      await database.sync();                /*.sync vai verificar os modelos neste projetos com
9                                             as tabelas no banco e garantir que existam ou estejam iguais*/
10
11
12  })();
```

The Explorer sidebar on the left shows the project structure:

- SEQUELIZE-DEMO
  - node\_modules
  - CRUD.sql
  - db.js
  - index.js
  - package-lock.json
  - package.json
  - produto.js

The Terminal at the bottom shows the execution of the command `node ./index.js` and the resulting SQL queries:

```
PS C:\Users\nadja\OneDrive\Documents\softex\sequelize-demo> node ./index.js
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'produtos' AND TABLE_SCHEMA = 'crud'
Executing (default): CREATE TABLE IF NOT EXISTS `produtos` (`id` INTEGER NOT NULL , `nome` VARCHAR(255) NOT NULL, `preco` DECIMAL, `descricao` VARCHAR(255), `createdAt` DATETIME NOT NULL, `updatedAt` DATETIME NOT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB;
Executing (default): SHOW INDEX FROM `produtos`
PS C:\Users\nadja\OneDrive\Documents\softex\sequelize-demo>
```

★ Ao executar o código acima, a tabela foi criada

The screenshot shows the MySQL Workbench interface. On the left, the 'Navigator' pane displays the 'produtos' table structure with columns: id (int PK), nome (varchar(255)), preco (decimal(10,0)), descricao (varchar(255)), createdAt (datetime), and updatedAt (datetime). The 'Information' pane shows the table's columns and their data types. The 'Query' pane contains the SQL statement: `SELECT * FROM crud.produtos;`. The 'Result Grid' shows the query results, which are currently empty. The 'Output' pane shows the execution details: 1 row(s) returned, 0.015 sec / 0.000 sec.

Table: **produtos**

Columns:

- id: int PK
- nome: varchar(255)
- preco: decimal(10,0)
- descricao: varchar(255)
- createdAt: datetime
- updatedAt: datetime

Query: `SELECT * FROM crud.produtos;`

Result Grid:

id	nome	preco	descricao	createdAt	updatedAt
----	------	-------	-----------	-----------	-----------

Output:

#	Time	Action	Message	Duration / Fetch
1	15:13:40	SELECT * FROM crud.produtos LIMIT 0, 1000	0 row(s) returned	0.015 sec / 0.000 sec

★ Criei um produto no arquivo ./index e executei.

The screenshot shows the Visual Studio Code interface with the following components:

- EXPLORER:** Shows the project structure for 'SEQUELIZE-DEMO', including files like 'CRUD.sql', 'db.js', 'index.js', 'package-lock.json', 'package.json', and 'produto.js'.
- EDITOR:** Displays the 'index.js' file with the following JavaScript code:

```
const Produto = require ('./produto');           //carregando os modelos em memória

await database.sync();                             /*.sync vai verificar os modelos neste projetos com
                                                    as tabelas no banco e garantir que existam ou estejam iguais*/

const novoProduto = await Produto.create({
  id: 32,
  nome: 'Teclado',
  preco: 389,
  descricao: 'Teclado Mecânico'
})
console.log(novoProduto);

})();
```
- TERMINAL:** Shows the execution of the command `node ./index.js` and the resulting database operations:

```
PS C:\Users\nadja\OneDrive\Documents\softex\sequelize-demo> node ./index.js
Executing (default): SELECT TABLE_NAME FROM INFORMATION_SCHEMA.TABLES WHERE TABLE_TYPE = 'BASE TABLE' AND TABLE_NAME = 'produtos' AND TABLE_SCHEMA = 'crud'
Executing (default): SHOW INDEX FROM `produtos`
Executing (default): INSERT INTO `produtos` (`id`,`nome`,`preco`,`descricao`,`createdAt`,`updatedAt`) VALUES (?, ?, ?, ?, ?, ?);
produto {
  dataValues: {
    id: 32,
    nome: 'Teclado',
    preco: 389,
    descricao: 'Teclado Mecânico',
    updatedAt: 2023-10-12T18:34:03.763Z,
    createdAt: 2023-10-12T18:34:03.763Z
  },
  _previousDataValues: {
    id: 32,
    nome: 'Teclado',
    preco: 389,
    descricao: 'Teclado Mecânico',
    createdAt: 2023-10-12T18:34:03.763Z,
```
- STATUS BAR:** Shows the current file is 'index.js' at line 18, column 6, using UTF-8 encoding and CRLF line endings.

★ Produto adicionado na tabela.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

crud

Tables

produtos

Columns

- id
- nome
- preco
- descricao
- createdAt
- updatedAt

Indexes

Foreign Keys

Triggers

Administration Schemas

Information

Schema: crud

cadastro simples CRUD\* produtos produtos

Limit to 1000 rows

1 • SELECT \* FROM crud.produtos;

Result Grid

	id	nome	preco	descricao	createdAt	updatedAt
▶	32	Teclado	389	Teclado Mecânico	2023-10-12 18:34:03	2023-10-12 18:34:03
*	NULL	NULL	NULL	NULL	NULL	NULL

produtos 6

Apply Revert

Output

Action Output

#	Time	Action	Message	Duration / Fetch
✓ 1	15:13:40	SELECT * FROM crud.produtos LIMIT 0, 1000	0 row(s) returned	0.015 sec / 0.000 sec
✓ 2	15:29:09	SELECT * FROM crud.produtos LIMIT 0, 1000	0 row(s) returned	0.000 sec / 0.000 sec