

Avaliação Final Web-II – Acesso a Dados – CRUD - Mysql – ORM

Data: 24/11/2021

Aluno: Antônio Clementino Neto

Projeto: 05 – Livros x Editora

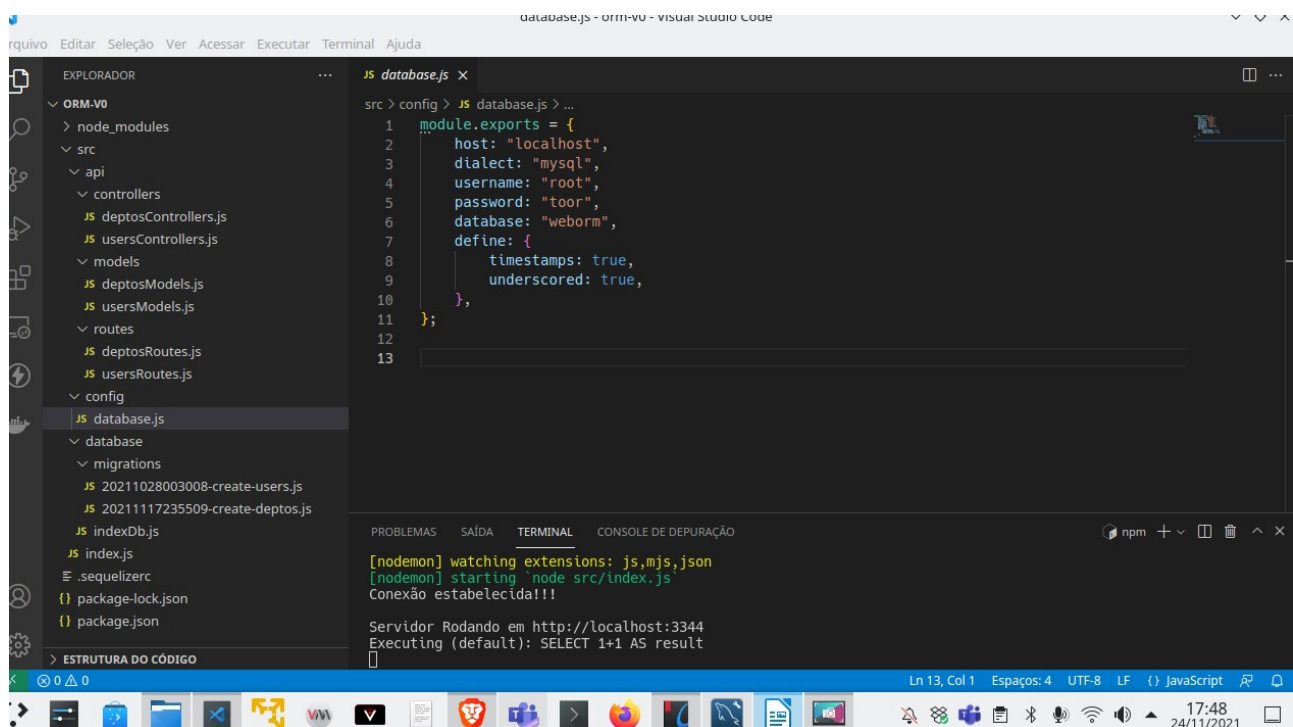
- Para a comprovação do desenvolvimento do projeto o aluno apresentará alguns prints obrigatórios do projeto iniciando da estrutura das tabelas

Especificação da Entidade – Tabela: EDITORA - EDI				
#	Tipo	Nome	<->	Descrição do campo
PK	inteiro	edi_codigo		Chave primária da tabela
	varchar	edi_nome	30	Nome da editora
	varchar	edi_cidade	20	Cidade da editora
	char	edi_estado	2	Estado – Unidade da federação
	inteiro	edi_fundacao		Ano de fundação da editora

Especificação da Entidade – Tabela: LIVRO - LIV				
#	Tipo	Nome	<->	Descrição do campo
PK	inteiro	liv_codigo		Chave primária da tabela
	varchar	liv_nomelivro	30	Nome do livro
	varchar	liv_autor	20	Nome do autor
	varchar	liv_categoria		Descrição da categoria-ciências sociais, informática...
	inteiro	liv_paginas		Número de páginas do livro
	inteiro	liv_anopublicacao		Ano de publicação do livro
FK	inteiro	edi_codigo		Código da Editora – chave estrangeira

1) Figura 1: Estrutura das tabelas – ao lado

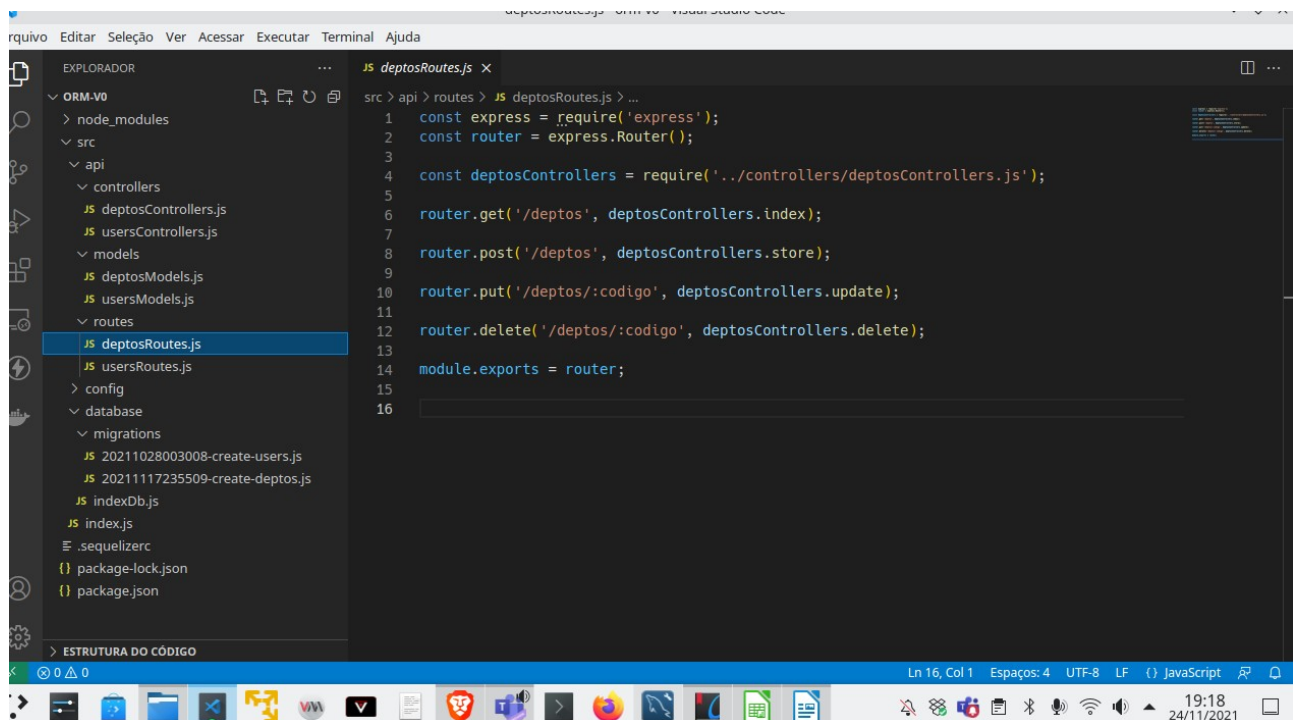
2) Figura 2: imagem da área de desenvolvimento do projeto (**Visual Studio Code**) com a estrutura de pastas a esquerda todas abertas com o arquivo **database.js** em destaque, este arquivo fica localizado na pasta **config**. É importante que o rodapé da tela esteja visível na imagem.



3) A partir de agora serão apresentados os print's sequenciais da primeira tabela informada no documento da tarefa. Neste exemplo apresentaremos a sequência dos códigos na seguinte ordem:

- routes
- controllers
- models
- migrations
- workbench

3.1) Figura 3: Routes.js da tabela departamentos – deptos



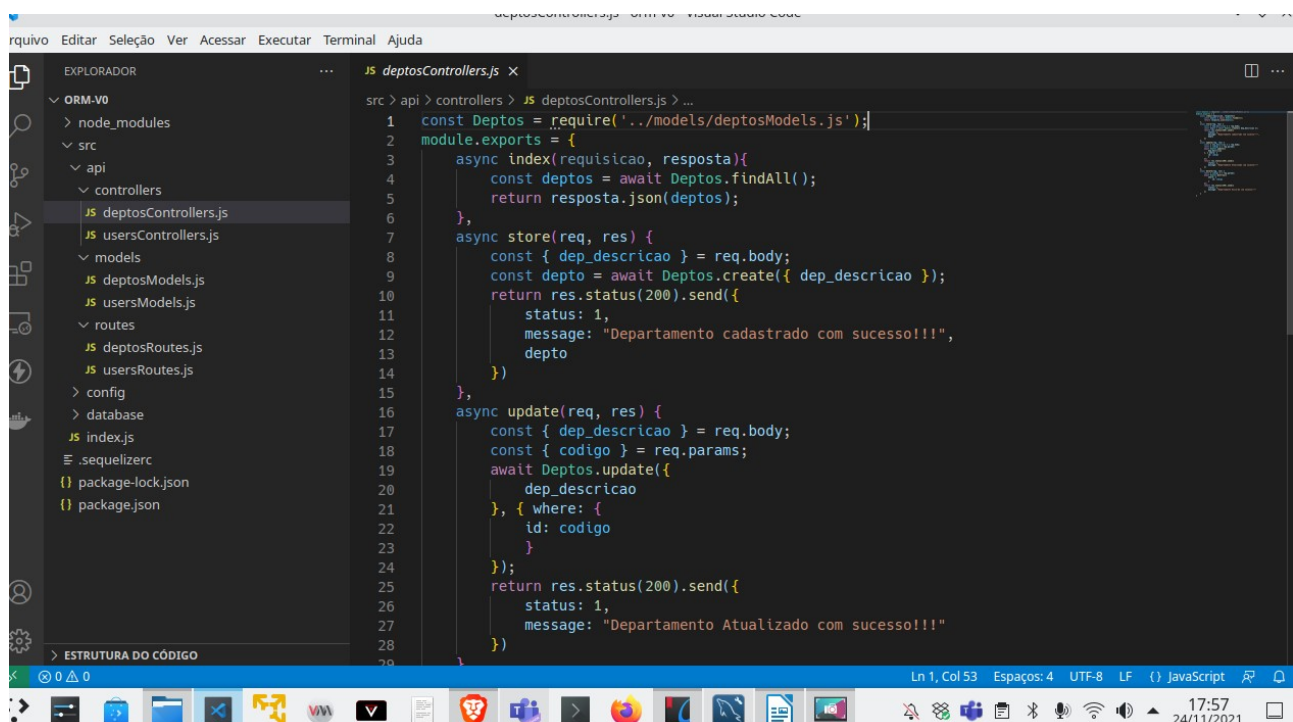
The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the following folders and files:

- ORM-V0
 - node_modules
 - src
 - api
 - controllers
 - JS deptosControllers.js
 - JS usersControllers.js
 - models
 - JS deptosModels.js
 - JS usersModels.js
 - routes
 - JS deptosRoutes.js (selected)
 - JS usersRoutes.js
 - config
 - database
 - migrations
 - JS 20211028003008-create-users.js
 - JS 20211117235509-create-deptos.js
 - indexDb.js
 - index.js
 - .sequelizerc
 - package-lock.json
 - package.json

The code editor shows the content of `JS deptosRoutes.js`:

```
1 const express = require('express');
2 const router = express.Router();
3
4 const deptosControllers = require('../controllers/deptosControllers.js');
5
6 router.get('/deptos', deptosControllers.index);
7
8 router.post('/deptos', deptosControllers.store);
9
10 router.put('/deptos/:codigo', deptosControllers.update);
11
12 router.delete('/deptos/:codigo', deptosControllers.delete);
13
14 module.exports = router;
15
16
```

3.2) Figura 4: Controllers da tabela departamentos - deptos



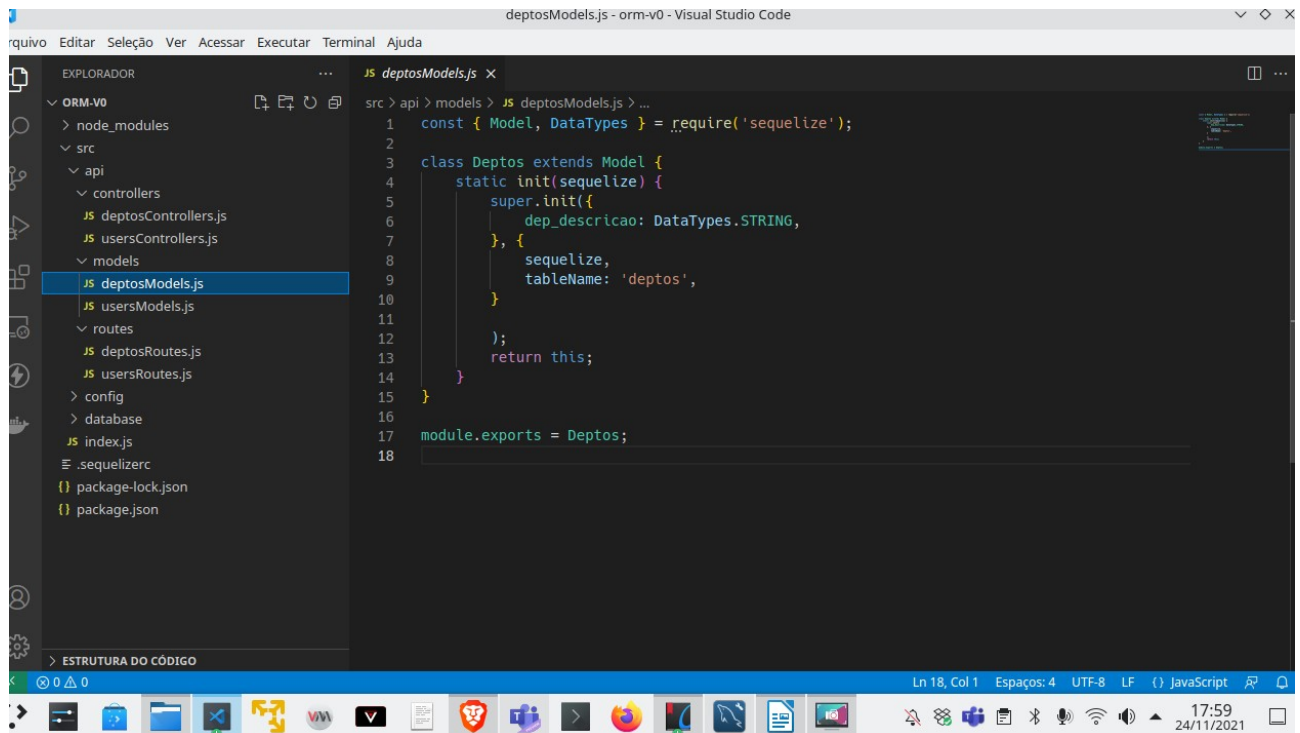
The screenshot shows the Visual Studio Code editor with the file explorer on the left and the code editor on the right. The file explorer shows the project structure with the following folders and files:

- ORM-V0
 - node_modules
 - src
 - api
 - controllers
 - JS deptosControllers.js (selected)
 - JS usersControllers.js
 - models
 - JS deptosModels.js
 - JS usersModels.js
 - routes
 - JS deptosRoutes.js
 - JS usersRoutes.js
 - config
 - database
 - index.js
 - .sequelizerc
 - package-lock.json
 - package.json

The code editor shows the content of `JS deptosControllers.js`:

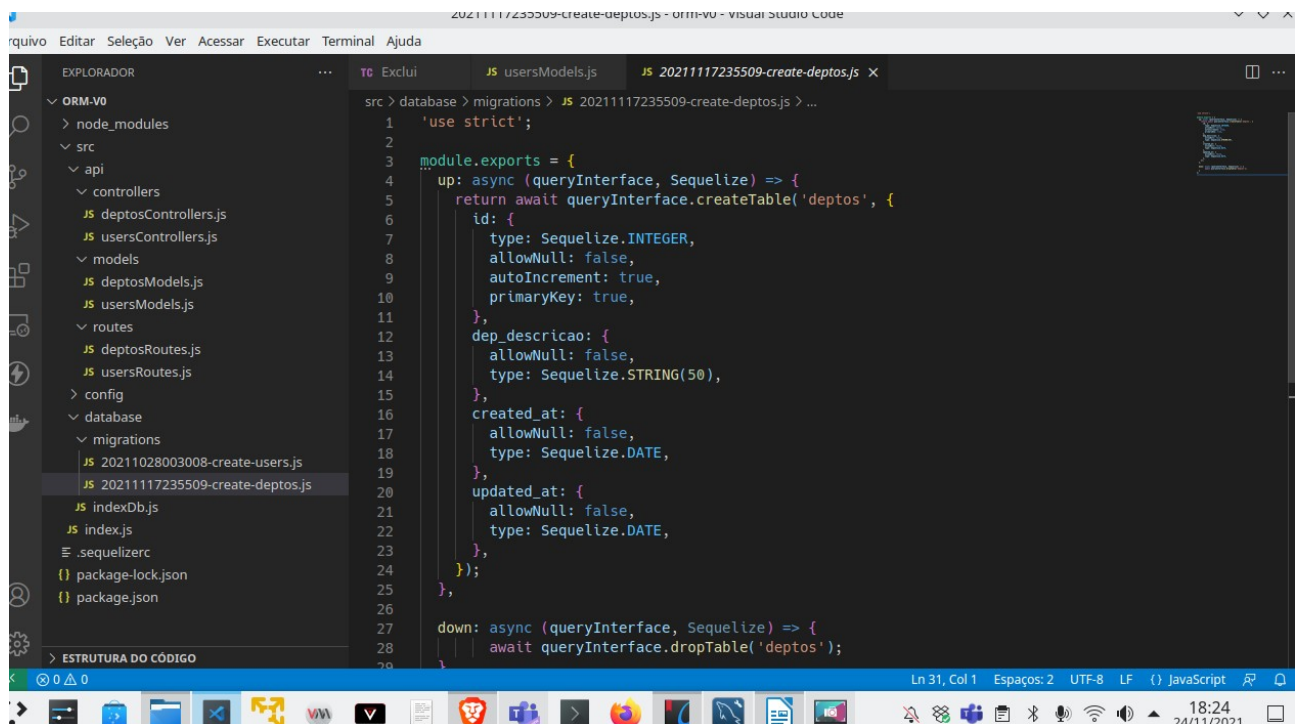
```
1 const Deptos = require('../models/deptosModels.js');
2 module.exports = {
3   async index(requisicao, resposta) {
4     const deptos = await Deptos.findAll();
5     return resposta.json(deptos);
6   },
7   async store(req, res) {
8     const { dep_descricao } = req.body;
9     const depto = await Deptos.create({ dep_descricao });
10    return res.status(200).send({
11      status: 1,
12      message: "Departamento cadastrado com sucesso!!!",
13      depto
14    });
15  },
16  async update(req, res) {
17    const { dep_descricao } = req.body;
18    const { codigo } = req.params;
19    await Deptos.update({
20      dep_descricao
21    }, { where: {
22      id: codigo
23    } });
24  });
25  return res.status(200).send({
26    status: 1,
27    message: "Departamento Atualizado com sucesso!!!"
28  });
29 }
```

3.3) Figura 5: Models da tabela departamentos - deptos



```
1 const { Model, DataTypes } = require('sequelize');
2
3 class Deptos extends Model {
4   static init(sequelize) {
5     super.init({
6       dep_descricao: DataTypes.STRING,
7     }, {
8       sequelize,
9       tableName: 'deptos',
10     });
11   }
12   return this;
13 }
14
15 module.exports = Deptos;
```

3.4) Figura 6: Migration departamentos

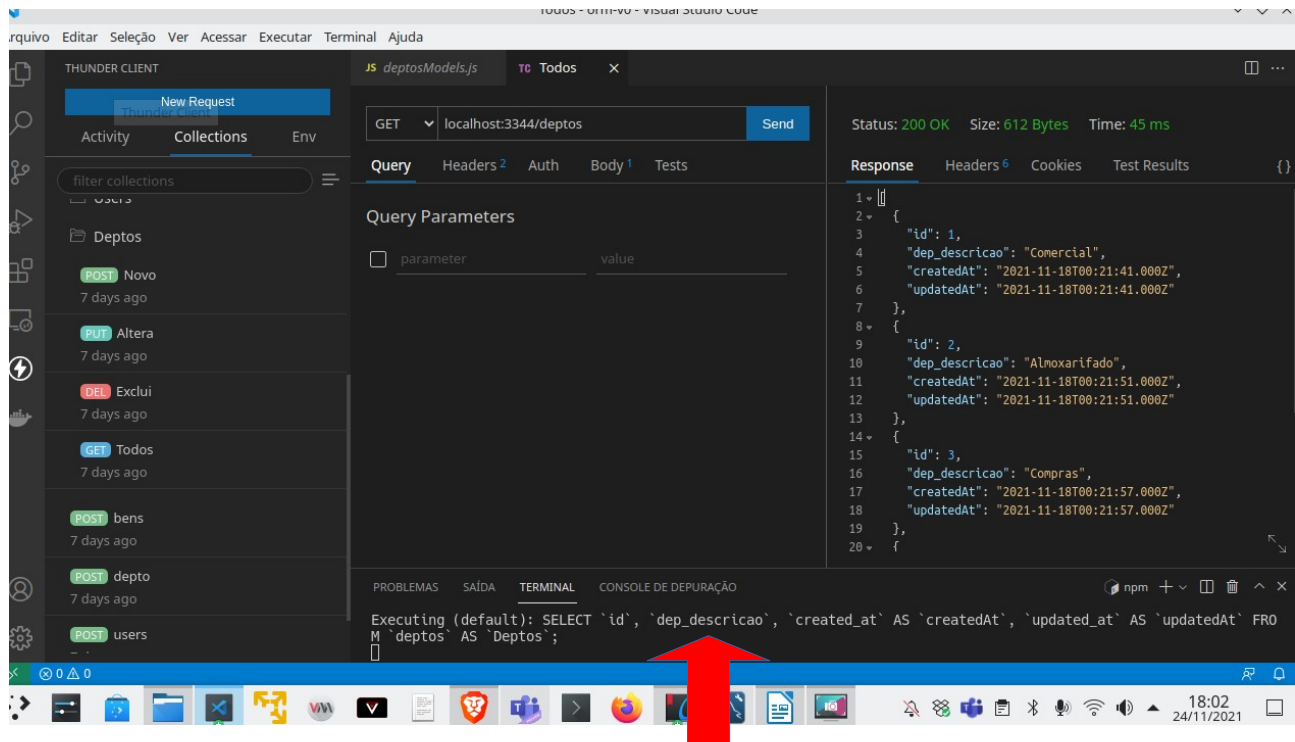


```
1 'use strict';
2
3 module.exports = {
4   up: async (queryInterface, Sequelize) => {
5     return await queryInterface.createTable('deptos', {
6       id: {
7         type: Sequelize.INTEGER,
8         allowNull: false,
9         autoIncrement: true,
10        primaryKey: true,
11      },
12      dep_descricao: {
13        allowNull: false,
14        type: Sequelize.STRING(50),
15      },
16      created_at: {
17        allowNull: false,
18        type: Sequelize.DATE,
19      },
20      updated_at: {
21        allowNull: false,
22        type: Sequelize.DATE,
23      },
24    });
25   },
26   down: async (queryInterface, Sequelize) => {
27     await queryInterface.dropTable('deptos');
28   }
29 }
```

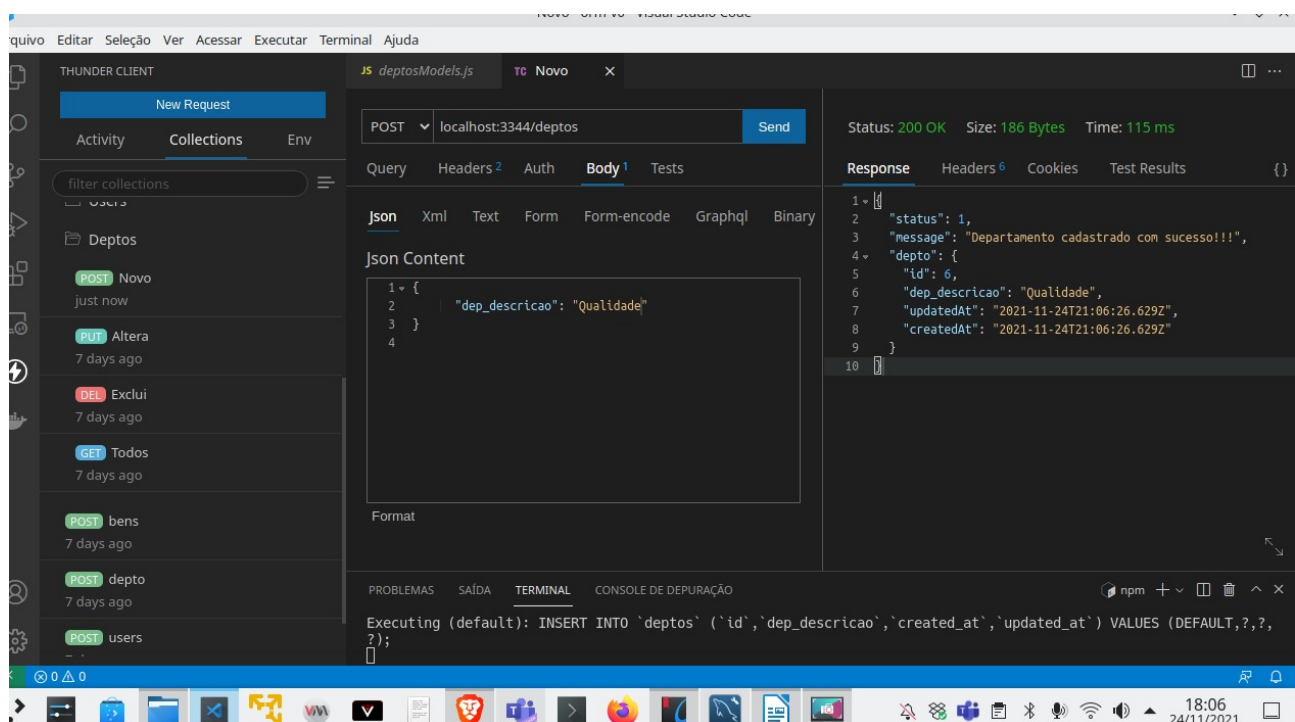
IMPORTANTE:

É necessário que as operações realizadas no backend sejam mostradas no terminal conforme pode ser vistas nas imagens a seguir abaixo da área do Thunder Client

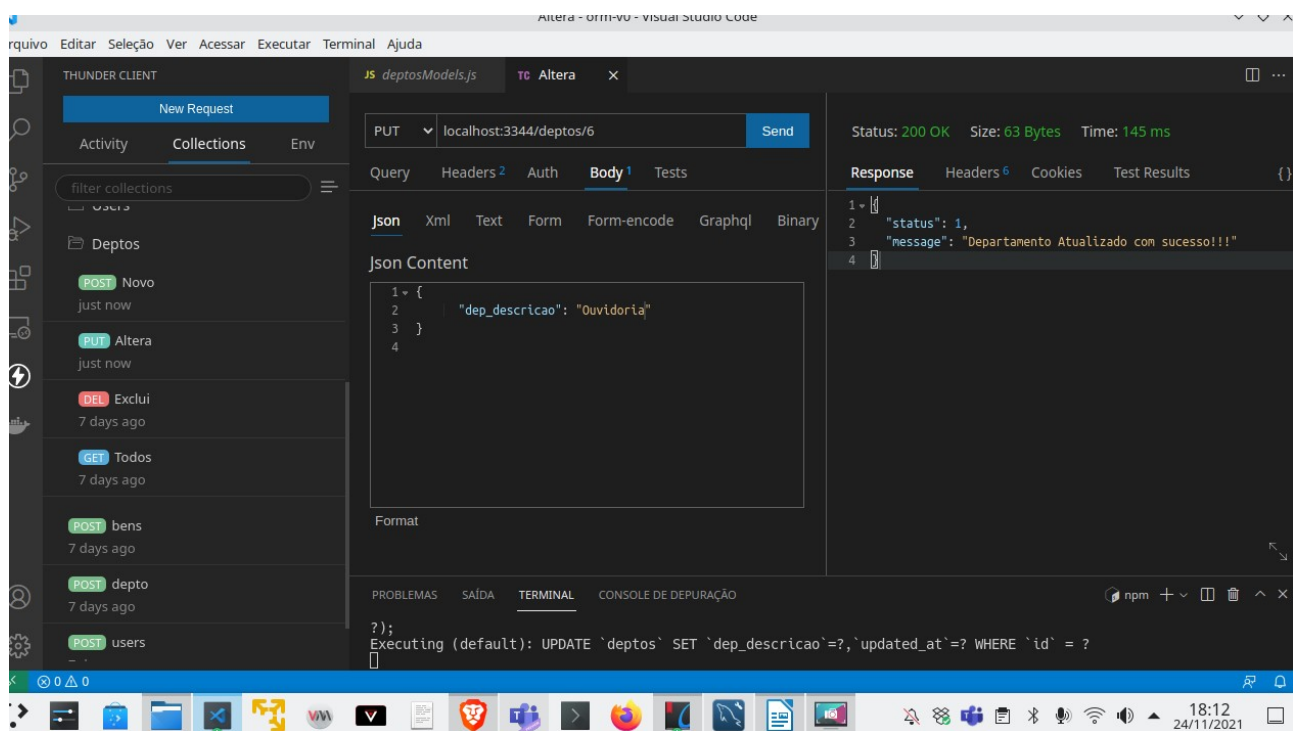
3.5) Figura 7: Protocolo GET método index.



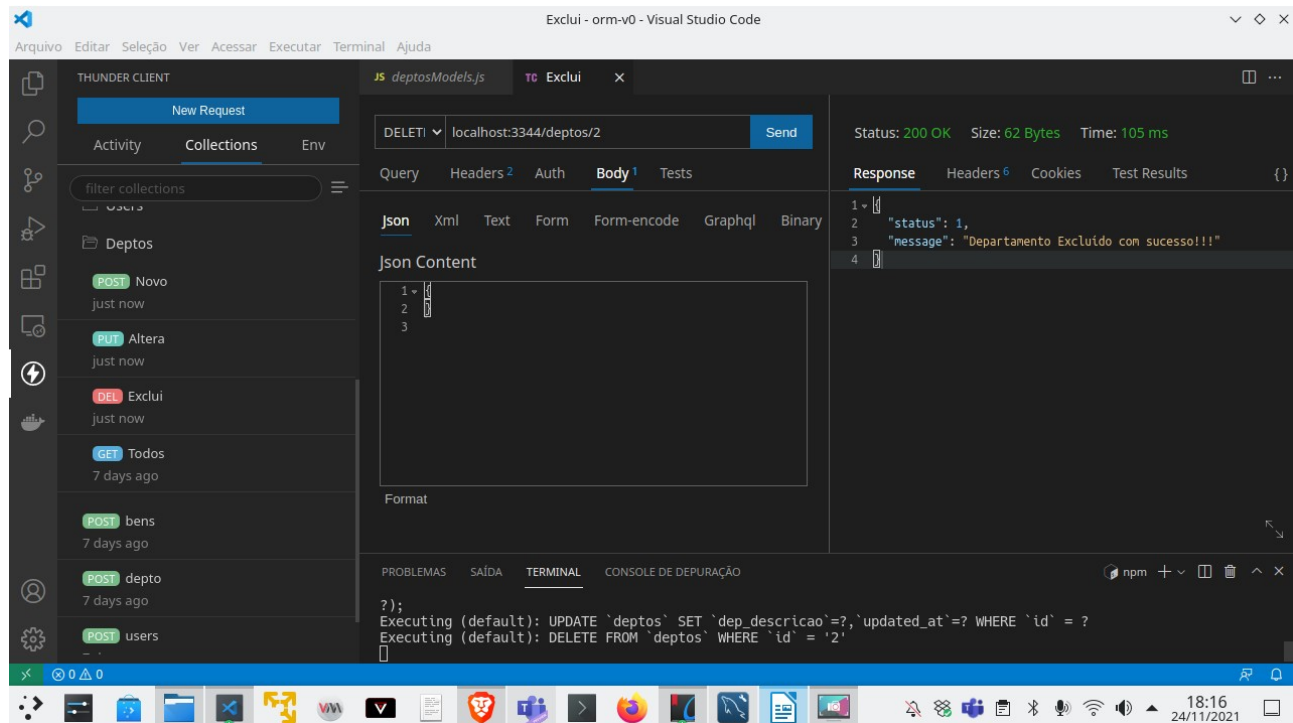
3.5) Figura 8: Protocolo POST método store (adicionar registro).



3.6) Figura 9: Protocolo PUT método update (alterar registro).



3.7) Figura 10: Protocolo DELETE método destroy (excluir registro).



3.8) Figura 11: Imagem do SGDB utilizado, mostrando o banco de dados à esquerda aberto listando os registros da tabela em questão (deptos).

