Modulo 07

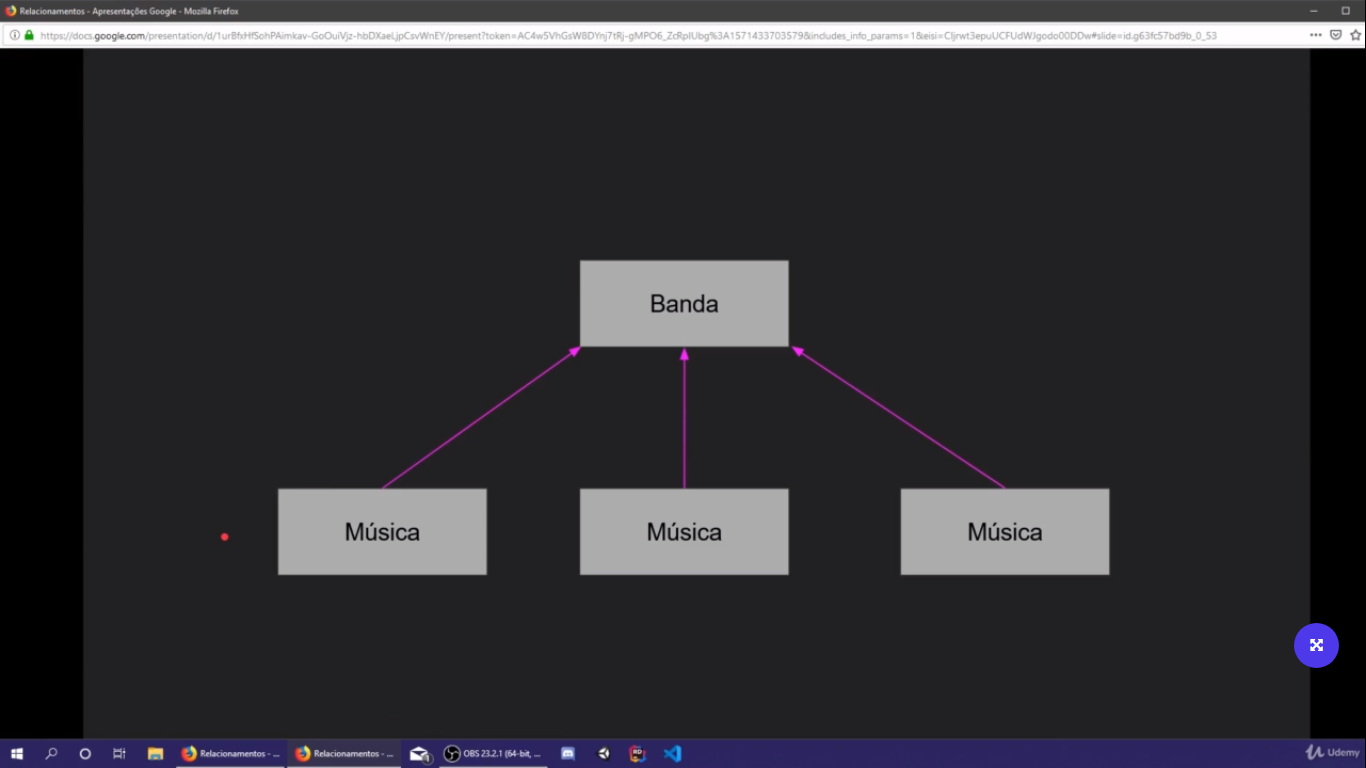
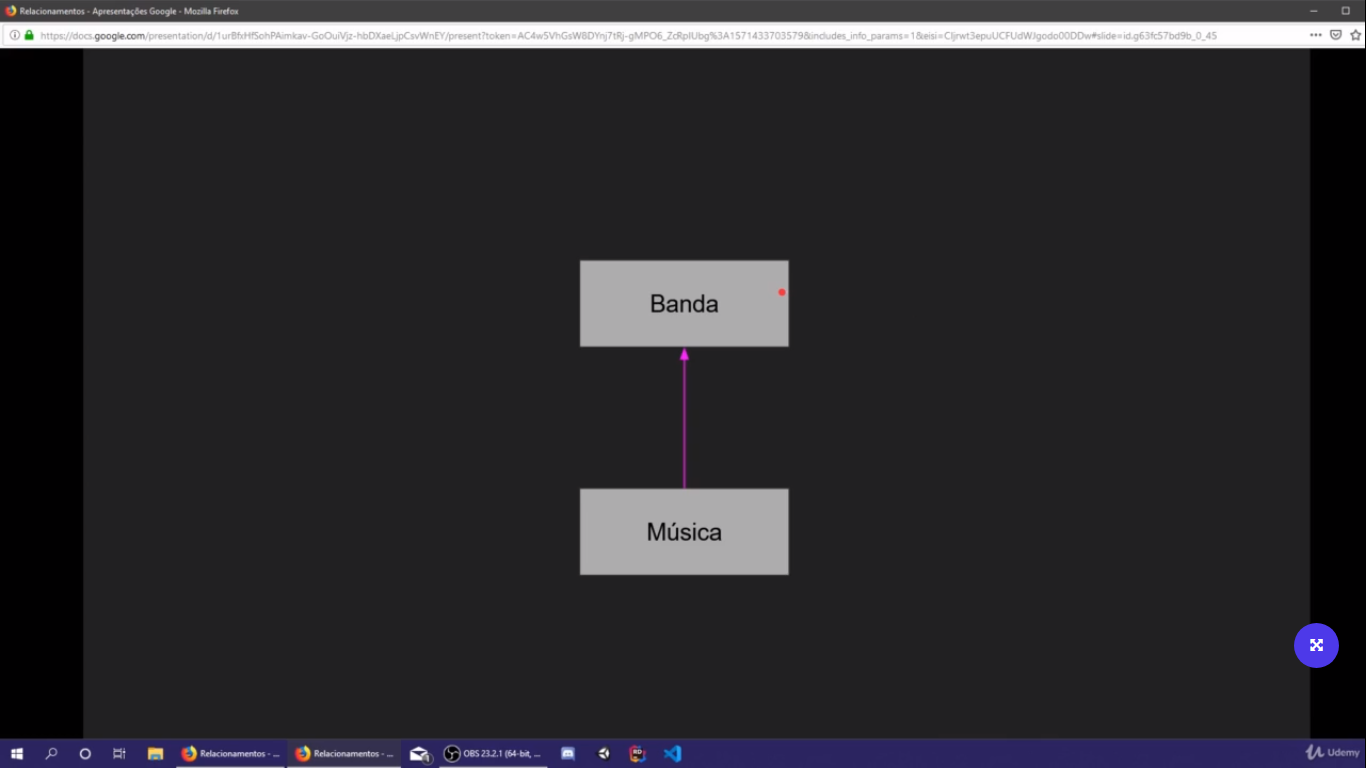
Projeto 02 Criando um blog com painel administrativo – CRUD completo

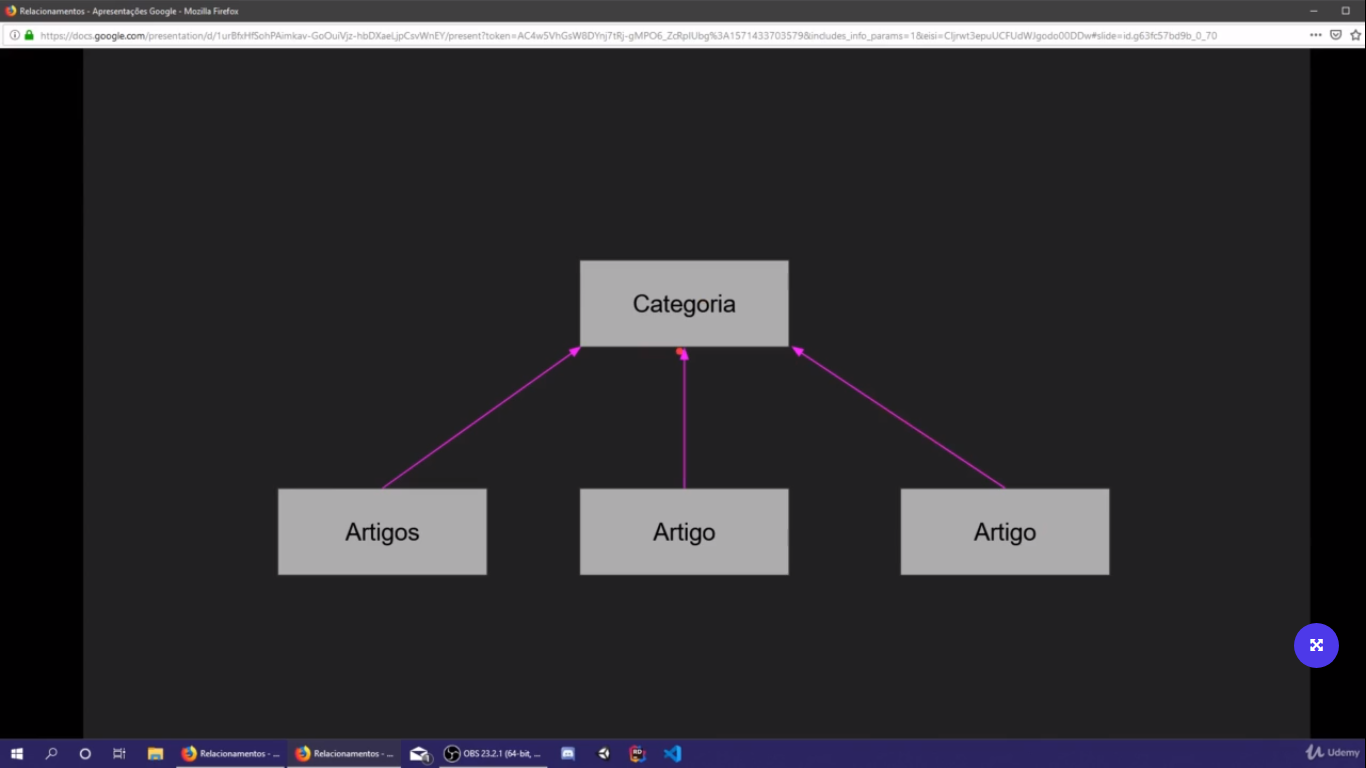
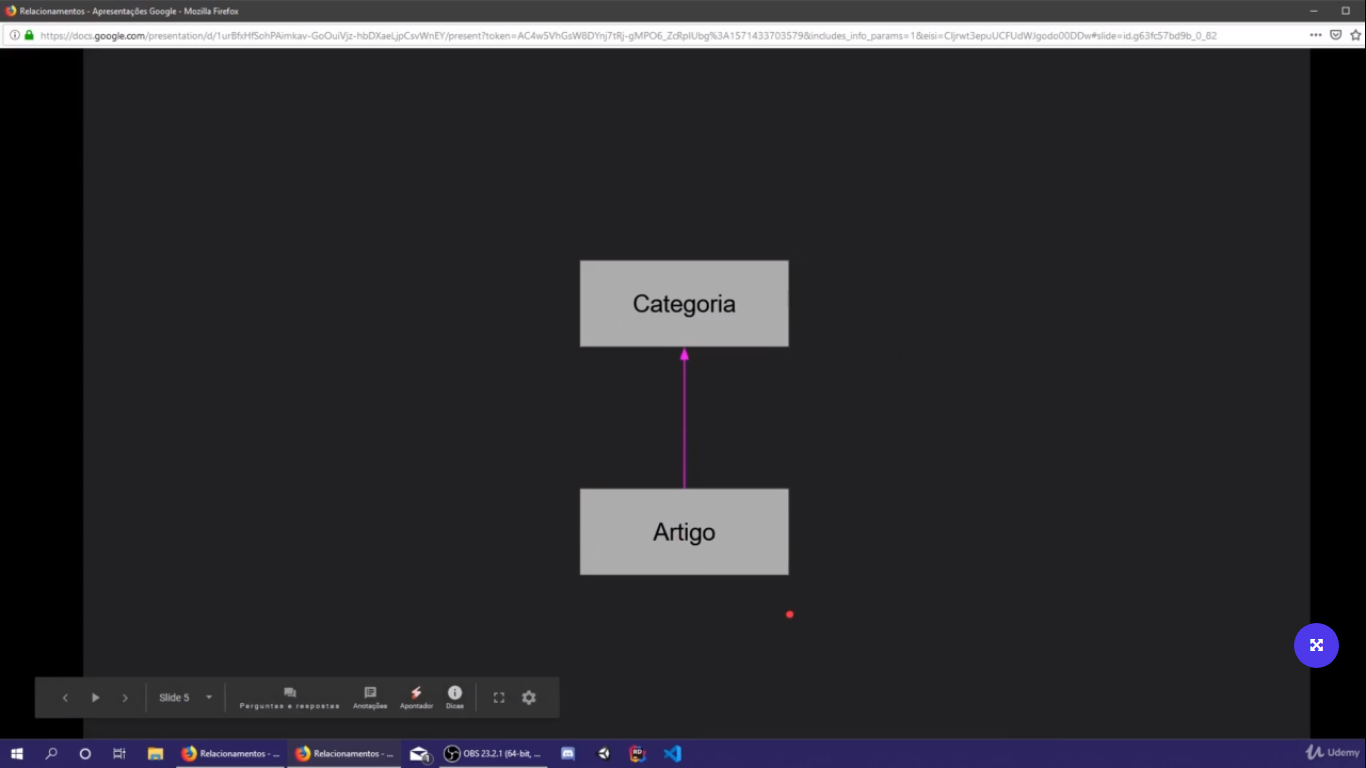
* Aula 01 Introdução

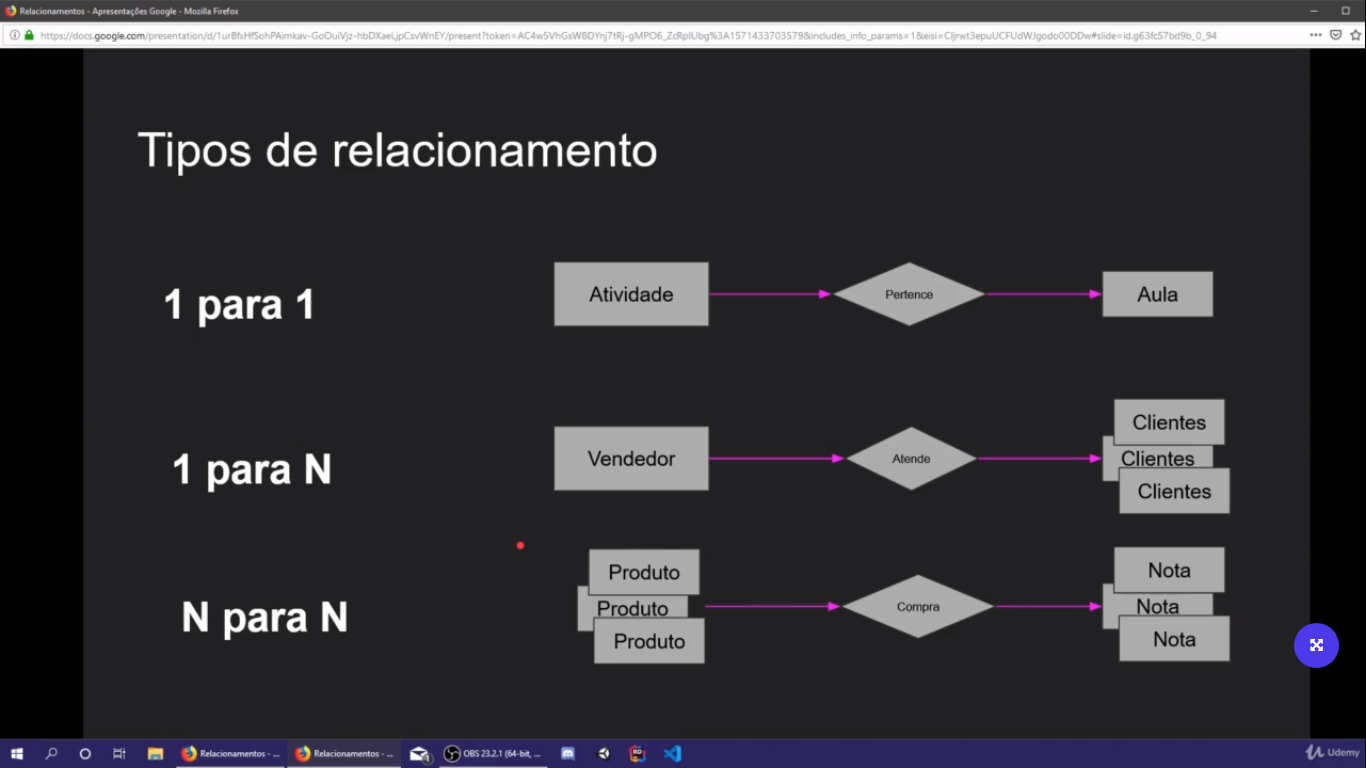
TinyMCE – Editor de texto mais utilizado para formatação de texto do estilo de blogs

* Aula 02 Relacionamento

Associação entre duas ou mais tabelas, ligar uma tabela conversar com uma outra







* Aula 03 Base do projeto

Ultilazaremos as bibliotecas, express, body-parser, ejs, sequelize e mysql2

Vamos configurar o projeto com bootstrap

Iremos criar o projeto na pasta guiaPress

Vamos começar!!!!

npm init -y

npm install --save express

npm install --save sequelize

npm install --save mysql2

npm install --save body-parser

npm install --save ejs

crie o arquivo index.js, e faça as configurações

index.js

const express = require('express')

const app = express()

app.get('/', (req, res) => {

    res.send('Bem vindo ao meu site')

})

app.listen(8080, () => {

    console.log('O Servidor está rodando na porta 8080');

})

Configure as:

- views engine(estrutura ‘html’ do projeto)

- body-parser(trabalha com formularios)

- Static(onde ficaram os arquivos estaticos)

- Criar seu banco de dados do projeto e suas conexões

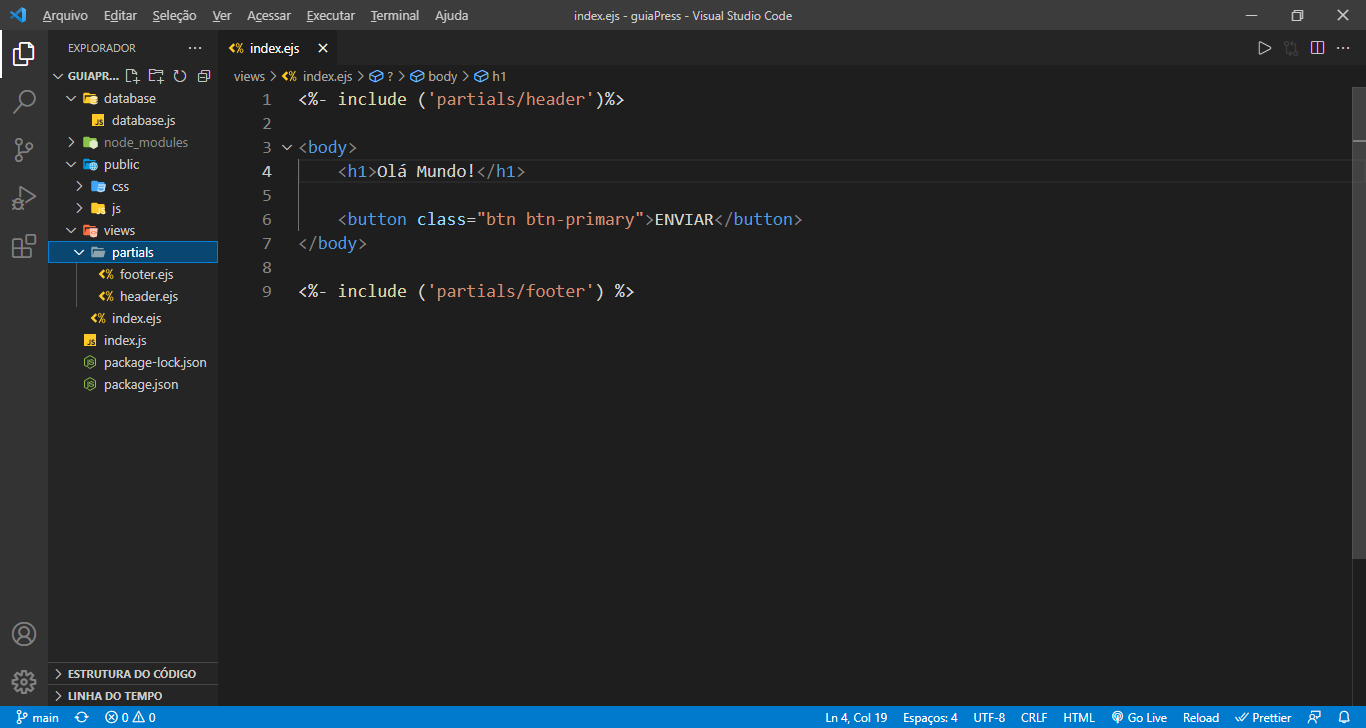
Feito essas configurações iremos instalar o bootstrap no projeto

Como estamos usando o ejs como o template engine, criamos arquivos parciais para o header e footer de nosso projeto

Na partials de header iremos fazer o link para nosso arquivo bootstrap.min.css, que baixamos e colocamos na pasta public/css

Já na partials footer, iremos colar os scripts do Jquery e Pooper que pegamos no site do bootstrap e por final lincar o script bootstrap.min.js que está na pasta public/js

Nosso projeto terá essa cara ao final da configurações iniciais:



- Banco de dados

Pasta database

- Arquivos publico/estáticos

Pasta public

- Visualização de paginas

Pasta views

-- dentro da pasta views temos nossas partials(header e footer)

- Arquivo principal do projeto

index.js

database.js

const Sequelize = require('sequelize')

const connection = new Sequelize('guiapress','root','123456',{

    host: 'localhost',

    dialect: 'mysql'

})

module.exports = connection

header.ejs

<!DOCTYPE html>

<html lang="pt-BR">

<head>

    <meta charset="UTF-8">

    <meta http-equiv="X-UA-Compatible" content="IE=edge">

    <meta name="viewport" content="width=device-width, initial-scale=1.0">

    <title>GuiaPress</title>

    <!-- Importando o Bootstrap -->

    <link rel="stylesheet" href="/css/bootstrap.min.css">

</head>

footer.ejs

<!-- Carregando Jquery -->

<script src="https://code.jquery.com/jquery-3.3.1.slim.min.js" integrity="sha384-q8i/X+965DzO0rT7abK41JStQIAqVgRVzpbzo5smXKp4YfRvH+8abtTE1Pi6jizo" crossorigin="anonymous"></script>

<!-- Carregando Popper -->

<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/popper.min.js" integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/l8WvCWPIPm49" crossorigin="anonymous"></script>

<!-- Carregando Bootstrap -->

<script src="/js/bootstrap.min.js"></script>

</html>

index.js

const express = require('express')

const app = express()

const bodyParser = require('body-parser')

const connection = require('./database/database')

// Viem Engine

app.set('view engine', 'ejs')

// Static

app.use(express.static('public'))

// Body-parser

app.use(bodyParser.urlencoded({extended: false}))

app.use(bodyParser.json())

// Database

connection.authenticate()

.then(() => {

    console.log('Conexão feita com sucesso');

})

.catch( err => {

    console.log('ERRO ao conectar ao BANCO DE DADOS '+ err);

})

app.get('/', (req, res) => {

    res.render('index')

})

app.listen(8080, () => {

    console.log('O Servidor está rodando na porta 8080');

})

* Aula 04 Arquitetura e organização de aplicação Node.js

O projeto e divido por dorminio, vamos usar um exemplo de um blog que tem dois domínios, o domínio de artigos e o domínio de categorias, são duas entidades diferentes, duas pastas diferentes

Controler -> nome do arquivo dado, onde ficara toda a logica da nossa aplicação, onde iram ficar nossas rotas

Todas as funcionalidades relacionadas a categoria do nosso projeto serão salvas dentro de scripts dentro dessa pasta que iremos criar com o nome de catogories

Iremos criar uma pasta chamada de articles, onde ficaram as logicas dos artigos

Agora para definir a rotas em um arquivo diferente ao arquivo principal iremos usar um recurso do express chamado express routers, que permite que você crie rotas sem a necessidade de trabalhar com a variável app

Na pasta categories iremos criar um arquivo CategoriesController.js

CategoriesController.js

const express = require('express')

const router = express.Router()

router.get('/categories', (req, res) => {

    res.send('Rota de categorias')

})

router.get('/admin/categories/new', (req, res) => {

    res.send('Rota para criar uma nova categoria!')

})

module.exports = router

e iremos linkar ao nosso arquivo principal

index.js

const CategoriesController =

require('./categories/CategoriesController')

app.use( '/' , CategoriesController)

agora toda vez que acessamos nosso projeto que tenha o caminho indicado ele ira buscar na nossa nova variável que recebe as routes de outro arquivos

prefixo, e algo que vai somar a essa rota passada.

Se nesse prefixo tivesse ‘/antes’, as rotas no navegador terias que se inicidadas com /antes e o caminho de uma rota

<http://localhost:8080/antes/categories>

* Aula 05 Model de categoria