

* Aula 2 como Instalar o nodeJs

Baixe pelo site:

<https://nodejs.org/en/>

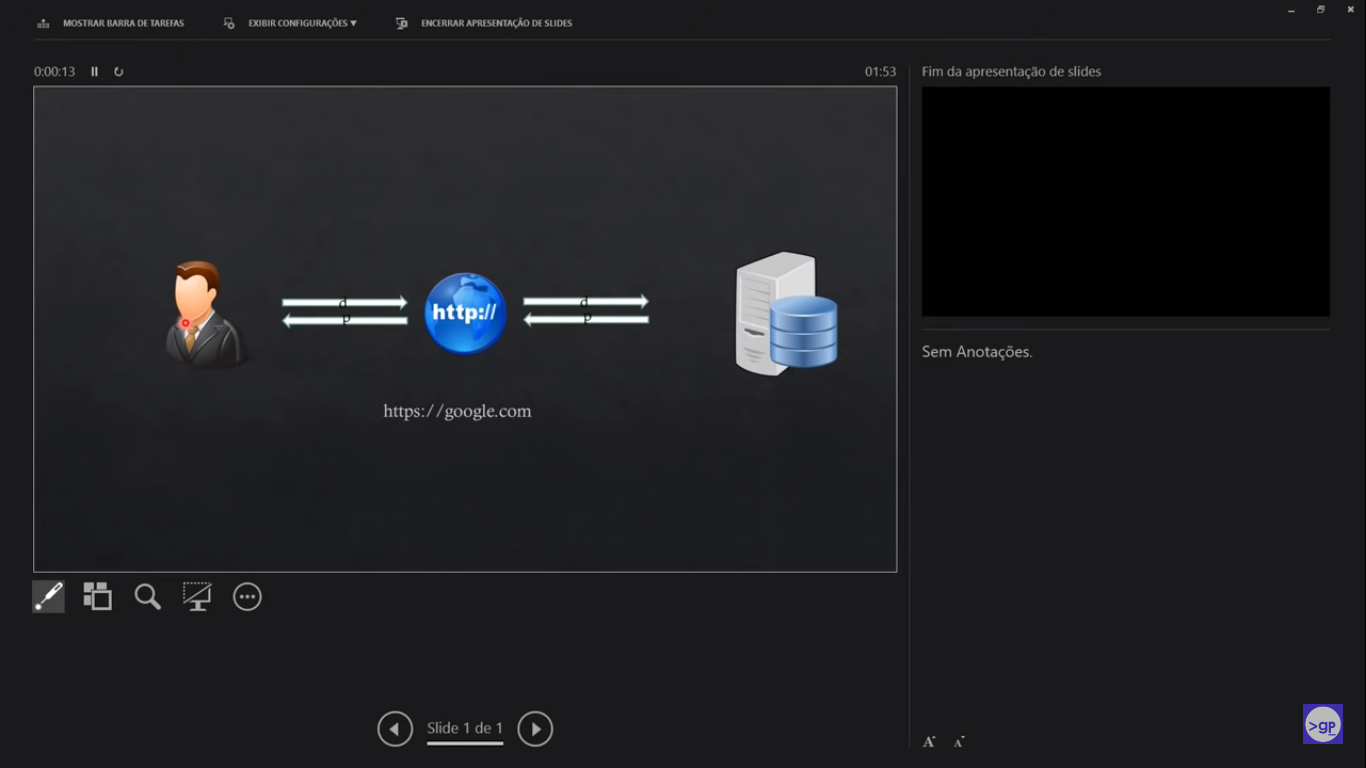
LTS - Opção principal livre de bugs

Currents - Opção para testar novas funcionalidades

node -v – verifica qual a versão instala na sua maquina

npm -v – verifca sua versão do npm

* Aula 5 protocolo http



let http = require('http')

// Para criar um servidor

http.createServer(function(req, res){

    res.end('Hello world')

}).listen(8081)

console.log('servidor rodando');

// http://localhost:8081/

// servidor aberto na porta 8081

// a cada alteração e nescessário fechar o servidor

// e rodar novamente

// ctrl + c  -> para o servidor

// no code runner ctrl + Alt + m

* Aula 6 Introdução ao Express

Para instalar o express no seu projeto:

npm install express --save

* Aula 7 Rotas

const express = require('express')

const app = express()

// Rotas - caminhos para a sua aplicação

app.get('/', function(req, res){

    res.send('Seja bem vindo ao meu app')

})

app.get('/sobre', function(req, res){

    res.send('minha página sobre')

})

app.get('/blog', function(req, res){

    res.send('Bem vindo ao meu Blog')

})

app.listen(8081, function(){

    console.log('servidor rodando na url http://localhot:8081');

})

// localhost:8081

// para deixar seu servidor rodando

// \*\* IMPORTANTE \*\* - sempre deixar essa

// linha por ultimo, ao final do código

* Aula 8 Parâmetros

app.get('/ola/:nome/:profissao', function(req, res){

    res.send(req.params)

})

// http://localhost:8082/ola/rodrigo/programador

// {"nome":"rodrigo","profissao":"programador"}

app.get('/ola/:nome/:cargo/:cor', function(req, res){

    res.send(`Olá ${req.params.nome}, Seu cargo é ${req.params.cargo}, Sua cor favorita é: ${req.params.cor}`);

})

* Aula 9 Nodemon

Digite o código:

npm install nodemon -g

nodemon app.js no terminal, agora seu servidor ficará atualizando sozinho

* Aula 10 exibindo HTML

Crie uma pasta de html no seu projeto e crie um arquivo html chamado index.html

Agora do seu arquivo de app.js mude o diretório do send para sendFile, passando o caminho absoluto do arquivo

app.get('/', function(req, res){

    res.sendFile( \_\_dirname + '/html'+ '/index.html')

})

Agora teremos o nosso caminho de paginas

app.get('/', function(req, res){

    res.sendFile( \_\_dirname + '/html' + '/index.html')

})

app.get('/sobre', function(req, res){

    res.sendFile(\_\_dirname + '/html' + '/sobre.html')

})

app.get('/blog', function(req, res){

    res.sendFile(\_\_dirname + '/html' + '/blog.html')

})

* Instalando o mysql

Baixe o apk no site <https://dev.mysql.com/downloads/>

Procure pelo do download de Community e por mysqlCommunityServer

Escolha sua versão de sistema operacional, escolha a versão MySql Installer, que dará duas opções de arquivo um completo e outro online.

Agora vamos instalar:

Selecione server Only

Agora vamos configurar:

Escolha development computer, coloque uma senha de usuário root e finalize a instalação

Agora vamos linkar o mysql ao cmd

Disco local – arquivo de programas – mysql – mysqlserver – bin

Copie o link dessa pagina e vá em propriedade no meu computador com o botão direito – configurações avançado do sistema – aba avançado – variáveis do ambiente – clique em path – editar – novo – e por fim cole o link da pasta copiada e ok para salvar

Para entrar no mysql vá no arquivo mySql Command line client e digite o password do usuário root que foi cadastrado na instalação

Feito seu mysql estará pronto para uso

* Aula 12 criando tabela no mysql

Digite no prompt de comando:

mysql -h <nome\_do\_servidor> -u <usuario> -p

ficaria:

mysql -h localhost -u root -p

ao digitar isso ele pedirá a senha do root, ao digitar e confirma, você estará no mysql

SHOW DATABASES; -> mostra os bancos de dados presente no mysql

CREATE DATABASE <nome\_do\_banco>; -> cria um banco de dados

CREATE DATABASE sistemaDeCadastro;

USE <nome\_do\_banco>; -> para acessar o banco de dados

USE sistemaDeCadastro

Agora estamos dentro da tabela sistemaDeCadastro, todo os comandos farão referência a ela

SHOW TABLES; -> mostra as tabelas dentro do banco

CREATE TABLE <nome\_da\_tabela>(

<Nome> <tipo>

);

CREATE TABLE usuarios(

nome VARCHAR(50),

email VARCHAR(100),

idade INT

);

DESCRIBE <nome\_da\_tabela> -> para ver a estrutura da tabela

DESCRIBE usuarios

* Aula 13 manipulando o mysql

INSERT INTO <nome\_da\_tabela>(<quais\_campos>)

VALUES(

‘PARA\_TEXTO’

NUMERICOS

); -> inserir dados na tabela em suas ordem

INSERT INTO usuarios(nome, email, idade)

VALUES (

‘Rodrigo’,

‘email@teste.com’,

29

);

SELECT \* FROM <nome\_da\_tabela> -> mostra os dados da tabela

SELECT \* FROM usuarios

SELECT \* FROM <nome\_da\_tabela>

WHERE <coluna> <condição> -> para receber condições para mostrar valores

SELECT \* FROM usuarios

WHERE idade = 8

* Aula 14 deletando registros

DELETE FROM usuarios

WHERE nome = ‘Luiz’

Ira deletar da tabela todas as linhas onde o nome seja igual a Luiz, se não tivesse a condição Where todo os dados de usuarios iriam serem apagados

* Aula 15

UPDATE <nome\_da\_tabela>

SET <coluna> = <valor\_para\_alterar>

WHERE <coluna> <condição>

Altera o registro de um dado de uma tabela

UPDATE usuarios

SET nome = ‘Novo nome’

WHERE nome = ‘Lucas’

* Aula 16 Sequelize

Digite os código em seu projeto:

npm install sequelize --save

npm install mysql2 --save

* Aula 18 conectar o mysql

Para se conectar crie um arquivo js, test.js criado.

Digite esse código:

const Sequelize = require('sequelize')

const sequelize = new Sequelize('test', 'root', '123456', {

    host: 'localhost',

    dialect: 'mysql'

})

Nome do banco

Usuário

Senha

Em qual máquina está seu banco

Qual o banco de dados utilizado

sequelize.authenticate().then(function(){

    console.log('Conectado com sucesso');

    }

).catch(function(){

    console.log('Falha ao se conectar');

})

Para testar se está tudo certo, abra o terminal do seu projeto e entre com o código:

Nodemon test.js

* Aula 18 Models no Sequelize

Para criar sua tabela no seu banco use o código:

const Postagem = sequelize.define('postagens', {

    titulo: {

        type: Sequelize.STRING

    },

    conteudo:{

        type: Sequelize.TEXT

    }

})

Postagem.sync({force: true})

Nome da tabela

Coluna

Tipo

Para que seja criada a tabela

Agora vamos inserir o conteúdo da sua tabela:

Postagem.create({

    titulo: 'Um Título qualquer',

    conteudo: 'Um conteúdo qualquer para sua tabela'

})

Qual a tabela que foi criada junto com o a função create, passando como um objeto as colunas com seus valores respectivos.

No terminal chame pelo node test.js, pois caso esteja usando o nodemon test.js a cada alteração de letra ele irá inserir uma nova linha da sua coluna, por isso e importando não está com o servidor do nodemon ativado em seu terminal

* Aula 19 Handlebars

Instale essa template engine

npm install --save express-handlebars

agora vamos dizer para o express que queremos usar o handlebars como nosso template no nosso projeto:

// Config

    // Template Engine

    app.engine('handlebars', handlebars({defaultLayout: 'main'}))

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

agora precisamos criar a pasta de nosso layout

crie uma pasta views, dentro dela crie outra pasta layouts e dentro dessa pasta crie um arquivo chamado

main.handlebars

nesse arquivo handlebars crie sua pagina html e no body coloque

{{body}}

<body>

    {{{body}}}

</body>

Agora vamos configurar a conexão com o BD diretamente do nosso arquivo index.js

Recorte as linhas que criam a conexão e cole nesse arquivo index.js

// Config

    // Template Engine

    app.engine('handlebars', handlebars({defaultLayout: 'main'}))

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

    // Conexão com o banco de dados MySql

    const sequelize = new Sequelize('test', 'root','123456',{

          host: "localhost",

          dialect: 'mysql'

        })

Poderá agora apagar o arquivo test.js, que foi usado para criar o conteudo do BD

* Aula 20 criando um formulário

Crie um arquivo formulário.handlebars dentro da pasta views, e monte seu formulário normal

<form action="">

    <p>Título:</p>

    <input type="text" name="titulo">

    <p>Conteúdo</p>

    <textarea name="conteudo" id="" cols="30" rows="10"></textarea>

    <button type="submit">Cadastrar postagem</button>

</form>

Em seguida no arquivo index.js defina a rota para esse arquivo que foi criado com o uso do res.render passando o arquivo que ele deverá mostrar no navegador

// ROTAS

    app.get('/cad', function(req, res){

        res.render('formulario')

    })

Dessa maneira, quando no endereço for chamado o /cad irá aparecer o conteudo contido no arquivo formulario.handlebars

* Aula 21 enviando dados ao formulario

<form action="/umarotaqualquer" method="">

    <p>Título:</p>

    <input type="text" name="titulo">

    <p>Conteúdo</p>

    <textarea name="conteudo" id="" cols="30" rows="10"></textarea>

    <button type="submit">Cadastrar postagem</button>

</form>

A forma como o seu formulario vai ser enviado para o back-end, como os seus dados serão enviados.

Existem duas formas de envio o GET e POST

GET -> ele envia dados pela url

Ex: <http://localhost:8081/umarotaqualquer?titulo=victor&conteudo=lima>

Na maioria dos casos não e ultilizado o método GET, pois caso seja passado uma senha ela será exibida no url da página

POST -> ele passa os dados de outra forma e não pelo url

Agora vamos criar uma rota para receber esses dados de post n arquivo index.js

app.post('/add', function(req, res){

        res.send('Formulario recebido')

    })

É importante ressaltar o uso do app.post, fazendo isso estou dizendo que essa rota só vai ser acessada quando alguém fizer uma requisição do método POST, e nosso formulario faz uma requisição do método posto

Essa rota só e acessada pela requisição do tipo POST, mesmo que você digitar <http://localhost:8081/add> direto na url, ela não será acessada

Agora que temos o caminho feito, iremos usar uma biblioteca para pegar esses dados

Iremos utilizar o body parser que serve para receber dados de formulario do express

* Aula 22 Body parser

npm install body-parser --save

e no arquivo index.js carregue esse utilitário para poder usá-lo

const bodyParser = require('body-parser')

e coloque essas configurações

    // Body Parser

        app.use(express.urlencoded({extended:true}))

        app.use(express.json())

agora e só rodar o servidor nodemonindex.js

agora iremos pegar esses dados enviados pelo formulario

app.post('/add', function(req, res){

        let titulo = req.body.titulo

        let coneteudo = req.body.conteudo

        res.send('TEXTO: '+titulo +' Conteudo: '+coneteudo)

    })

Estamos utilizando o req.body.<nome\_do\_campo>, que foi definido e jogando em uma variavel

<form action="/add" method="POST">

    <p>Título:</p>

    <input type="text" name="titulo">

    <p>Conteúdo</p>

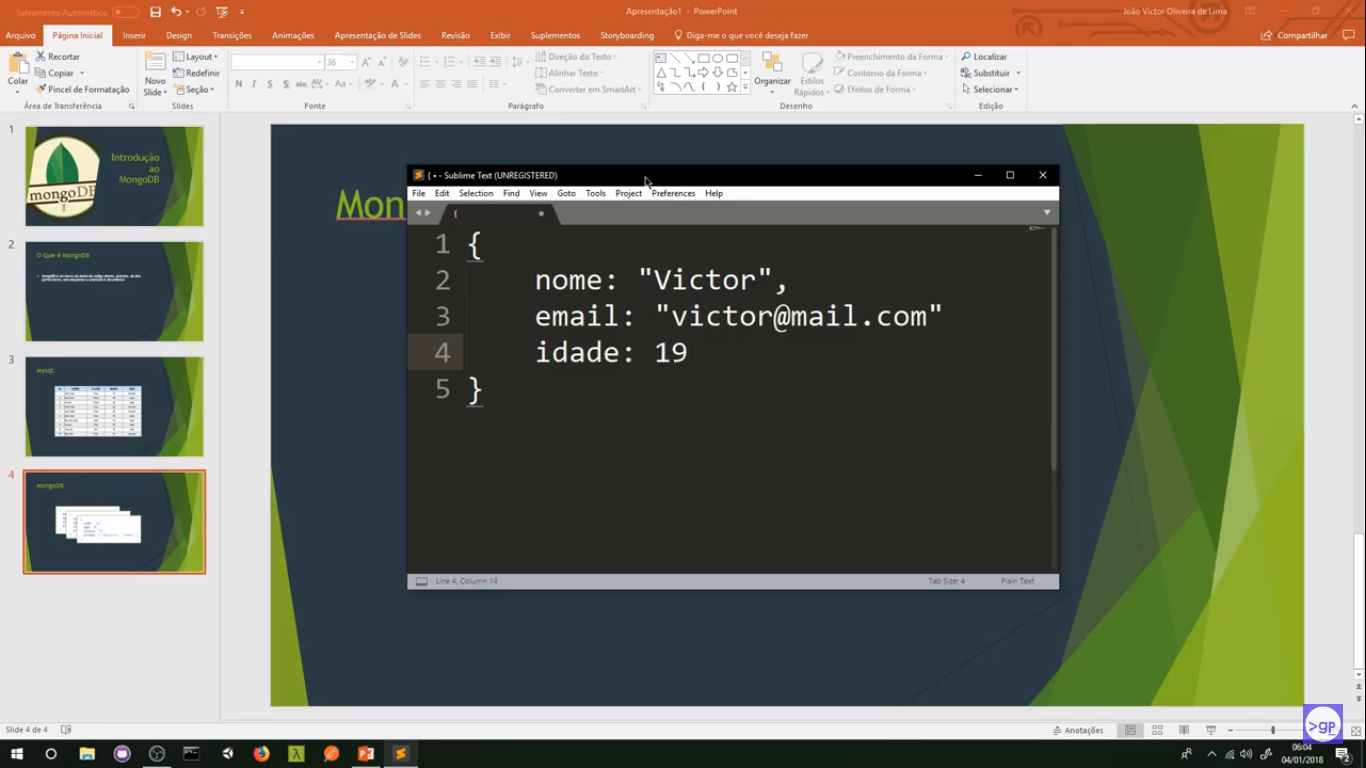
    <textarea name="conteudo" id="" cols="30" rows="10">

</textarea>

    <button type="submit">Cadastrar postagem</button>

</form>

* Aula 23 estruturando banco de dados
* Aula 27 introdução ao mongo DB

* Aula 28 instalando o mongo DB

<https://www.mongodb.com/try/download/community>

agora vamos fazer a configuração do mongo para acessar através da linha de comando

disco local -> arquivo de programa -> mondoDB -> server ->

3.6(versão instalada) -> bin

Copie o endereço dessa pagina, agora vamos registrar no path do nosso Windows

Meu computador -> botão direito -> propriedade -> configurações avançadas do sistema -> aba avançado -> variáveis de ambiente ->

Selecione path -> editar -> novo -> e cole o endereço copiado e salve e saia

Depois de feito isso vamos agora iniciar

Para começar a usar o mongo precisamos iniciar um servidor

Dentro do disco local, crie uma nova pasta chamada data, aqui ficará registrada todos o uso de meu mongoDB

Dentro da pasta data crie uma nova pasta chamada db

Cmd -> mongod

Importante tem que deixar o terminal que foi chamado o mongoDB ativo, caso feche o seu servidor irá cair

show collections

mostra as coleções no seu banco como as tabelas

* Aula 29 iniciando com o mongoose

Na pasta do seu projeto instale o mongoose

npm install mongoose --save

const mongoose = require('mongoose')

mongoose.connect('mongodb://localhost/umbamcoqualquer')

em qual maquina está o seu servidor

o nome do banco onde você que se conectar

caso não exista ele irá criar sozinho

para criar um banco direto no terminal, basta usar

use <nome\_do\_banco>

use listaBanco

arquivo aprendendo.js

const mongoose = require('mongoose')

// Configurando o Mongoose

    mongoose.Promise = global.Promise

    mongoose.connect('mongodb://localhost/umbamcoqualquer', { useNewUrlParser: true })

    .then(() => console.log('MongoDB conectado com sucesso'))

    .catch(erro => console.log('ERRO ao se conectar: '+erro))

Para ver se tudo está certo digite no prompt do projeto

node aprendendo.js

* Aula 30 trabalhando com o mongoose

Vamos criar nosso modelo de tabela

// Model - Usuários

// Definindo Model

    const UsuarioSchema = mongoose.Schema({

        nome: {

            type: String,

            require: true  // campo obrigatório

        },

        sobrenome: {

            type: String,

            require: true

        },

        email: {

            type: String,

            require: true

        },

        idade: {

            type: Number,

            require: true

        },

        pais: {

            type: String

        }

    })

// Collection

    mongoose.model('usuarios', UsuarioSchema)

como se chamará essa collection

a quem essa colection faz referência

Agora vamos criar nossos dados para a tabela

const Usuario = mongoose.model('usuarios')

new Usuario({

    nome:'Rodrigo',

    sobrenome: 'Souza',

    email: 'email@teste.com',

    idade: 29,

    pais: 'BRA'

}).save()

.then(() => console.log('Usuario salvo'))

.catch(err => console.log('Houve um erro: '+err))

Agora quando rodarno terminal esse arquivo sera criado uma nova collection

node aprendendo.js

para salvar

em outro terminal

mongo - abre o banco

show databases; - mostra os bancos

use aprendendo - seleciona o banco

show collections; - mostra as tabelas

db.usuarios.find() - mostra todos os dados da tabela

* Aula 31 iniciando projeto Express.js

Instalando os npm:

npm install express --save

npm install express-handlebars --save

npm install mongoose --save

npm install body-parser –save

npm i nodemon

estruturando o esqueleto da aplicação:

app.js - arquivo principal

pasta models

pastas routes

pasta views -> layouts -> main.handlebars

(estrutura html -> body {{{body}}})

app.js

// Carregando módulos

const express = require('express')

const handleBars = require('express-handlebars')

const bodyParser = require('body-parser')

// const mongoose = require('mongoose')

const app = express()

// Configurações

    // Body Parser

    app.user(bodyParser.urlencoded({extended:true}))

    app.use(bodyParser.json())

    // Handlebars

    app.engine('handlebars', handleBars({defaultLayout: 'main'}))

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

// Rotas

// Outros

const PORT = 8081

app.listen(PORT, () => console.log('Servidor Rodando'))

* Aula 32 grupo de rotas no express

Pasta routes -> admin.js

const express = require('express')

const router = express.Router()

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

    res.send('pagina principal do painel ADM')

})

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

    res.send('Página de post')

})

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

    res.send('Página de categorias')

})

module.exports = router

app.js

// Carregando módulos

const express = require('express')

const handleBars = require('express-handlebars')

const bodyParser = require('body-parser')

const app = express()

const admin = require('./routes/admin')

// Configurações

    // Body Parser

    app.use(express.urlencoded({extended:true}))

    app.use(express.json())

    // Handlebars

    app.engine('handlebars', handleBars({defaultLayout: 'main'}))

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

// Rotas

    app.get('/',  (req, res) => {

        res.send('<h1>Página principal da aplicação</h1>')

    })

    app.get('/posts', (req, res) => {

        res.send('Lista Posts')

    })

    app.use('/admin',admin) // uso de profixo admin, tem uma series de paginas

// Outros

const PORT = 8081   // Inicia o seridor

app.listen(PORT, () => console.log('Servidor Rodando na porta http://localhost:'+PORT+'/'))

* Aula 33 arquivos estáticos

Pasta public - onde ficará todos os arquivos

Public -> css

Public -> js

Public -> img