Links Uteis

O projeto passo a passo  
<https://github.com/paulosalvatore/ocean-jornada-fullstack-novembro-22>

Programa para compartilhar tela escrita (Tipo uma Lousa)  
<https://miro.com/app/board/uXjVPGdjF2Y=/>

Pacote Express, para rodar o projeto em HTTP  
<https://www.npmjs.com/package/express>

Para configurar o Prittier  
<https://www.youtube.com/watch?v=vFByu7-f858>

Outros

<https://www.youtube.com/watch?list=PLzFYWfRVHp5Jx-iNLxhUydXwnYPqmV2MU&v=F81k9w8gDJs&feature=youtu.be&ab_channel=PauloSalvatore>

Nodemon

<https://www.npmjs.com/package/nodemon>

Banco de dados, Hanking

<https://db-engines.com/en/ranking>

Comando MongoDB

<https://www.npmjs.com/package/mongodb>

Para verificar algumas coisas da função await

<https://www.youtube.com/watch?v=QMjZyQZMOSY&feature=youtu.be&ab_channel=PauloSalvatore>

Bloco de Notas que envia texto para outras pessoas pela URL

<https://wordpad.cc/>

Banco de dados Mongo, para colocar na nuvem

<https://www.mongodb.com/atlas/database>

Colocar endereço IP fixo

<https://www.noip.com/pt-BR>

Para fazer conversão NodeJS para JS

<https://create-react-app.dev/docs/getting-started>

<https://vitejs.dev/guide/>

(usaremos o vitejs)

O PACKGE.JSON é o arquivo mais importante do projeto. É o coração. Para chamar ele, vai em Terminal > New Terminal e digita **npm init -y**

Criar o **index.js**, clicando em novo e editando o nome para **index.js**

Começa a escrever os códigos e para verificar o que foi escrito, usa o TERMINAL para ver, escrevendo o comando **node index.js**

Para fazer interface gráfica, criação de servidor, etc, é necessário instalar o EXPRESS, cada projeto deve ser feita a instalação

Depois de fazer o npm init -y é possível instalar o express usando o comando **npm install express**

\*Caso você tenha pego algum projeto sem a pasta **node\_modules** é só chamar a **npm install** que a pasta será criada no projeto já existem

Quando o node index.js estiver rodando no terminal, para parar é só clicar no terminal e apertar CRTL + C

**Instalar o Thunder Client**

Para testar a aplicação BackEnd usamos o Thunder Client, para tanto, ir em EXTENSOES e digitar Thunder Client e instalar. Depois clicar no ícone RAIO do lado esquerdo e gerar **New Request** , nesse caso mudamos o endereço para http://localhost:3000

**Usar os operadores**

**Conceito:** REST é uma convenção para melhorar a arrumação

Diagrama

Descrição gerada automaticamente

Fazer a aplicação usando NOME e IMAGEM

Lista ou Arrays é a mesma coisa.

**Const** – Criar informações que não mudam, não pode ser alterado

**Let** – A informação pode ser mudada

Var – Permite ser alterada

O CONST permite colocar item dentro do cochetes, exemplo fazer um PUSH no CONST lista [1,2,3], colchete é o símbolo que insere uma nova lista.

Iremos instalar o NODEMON que reinicializa automaticamente a aplicação quando salva.

Podemos instalar de forma completa, na maquina, ou somente no projeto.

Iremos usar a instalação somente no projeto: **npm install --save-dev nodemon**

Criaremos o SCRIPT, então modificamos

  "scripts": {

    "test": "echo \"Error: no test specified\" && exit 1"

  },

Por

  "scripts": {

    "start": "node index.js",

    "dev": "nodemon index.js"

  },

Para rodar o script devemos digitar no terminal: **npm run dev** , esse Script deve ser iniciado toda vez que abrir o projeto para trabalhar.

Para encerrar a aplicação **npm run dev** é necessário clicar no terminal e clicar Ctrl+C duas vezes

**Tip**: Atl + shift + seta para baixo duplica a linha

Para usar o MongoDB precisa instalar no terminal do VSCODE

npm i mongodb

E colocar outras constantes no VSCODE

const { MongoClient, ObjectId } = require("mongodb")

fica entre:

const express = require('express')

const { MongoClient, ObjectId } = require("mongodb")

const app = express()

Depois é necessário colocar o endereço do banco de dados para fazer a conexão, então precisa configurar a rota

const url ="mongodb://localhost:27017";

Depois temos que declarar o nome do banco de dados exatamente igual ao criado

const BancoDadosNome = "Ocean\_Jornada\_Fullstack\_Novembro\_2022";

Agora é hora de conectar, e é necessário usar o comando connect dentro de uma CONST com qualquer nome, nesse caso iremos usar o nome cliente

const client = MongoClient.connect(url);

como leva tempo uma conexão com o banco de dados, pois é uma PROMISE, é necessário fazer um Async e Await, necessita criar uma função, que daremos o nome de main, pois será o principal e faremos um main() para executar. Obs: Para usar o Await é necessário fazer uma função Async

async function main() {

    const client = await MongoClient.connect(url);

}

main();

Para que tudo funcione dentro do Async, é necessário colocar todo o código dentro do main(), trazendo o fechamento para o final

}

main();

Para colocar na NUVEM

Entrar no site <https://www.mongodb.com/atlas/database>

Criar usuário e senha

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

Adm: Admin

Senha: GqB1qB34Drs9wrdt

Transfor de mongodb+srv://admin:<password>@cluster0.pdjfbuo.mongodb.net/?retryWrites=true&w=majority

para

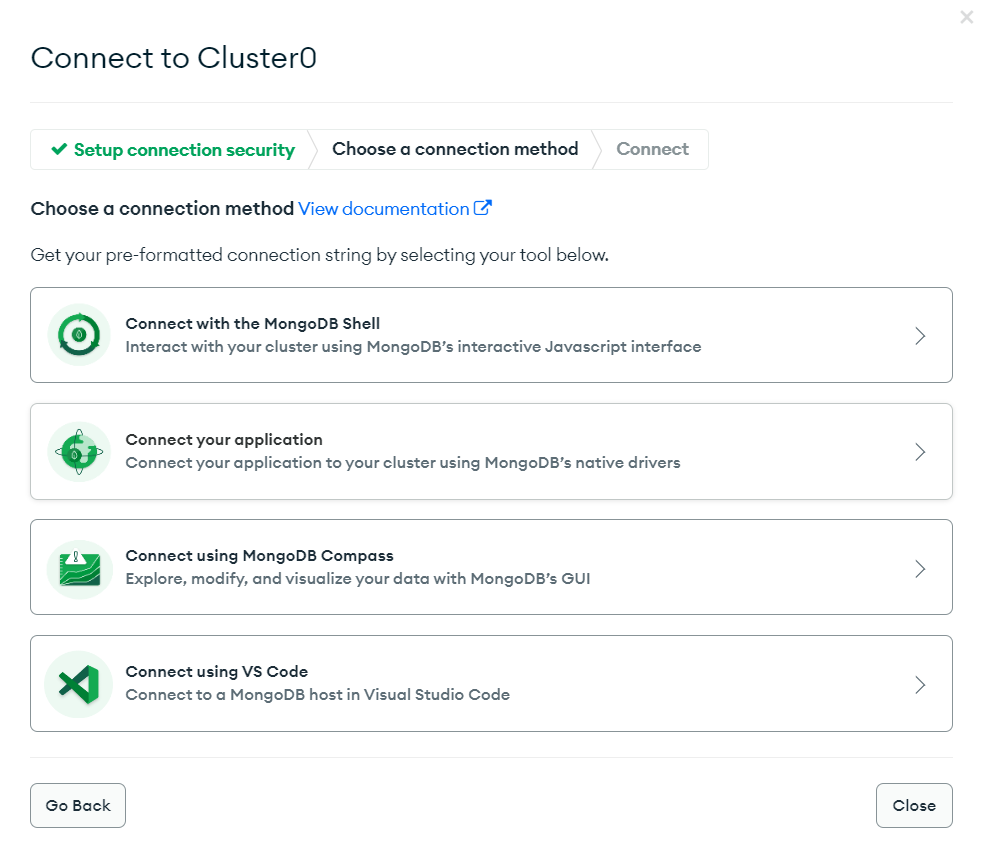
URL do BD:

mongodb+srv://admin:GqB1qB34Drs9wrdt@cluster0.pdjfbuo.mongodb.net/

Interface gráfica do usuário, Texto, Aplicativo, Email

Descrição gerada automaticamente

Conecta sua aplicação



Meu Git Desktop

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Faz uma conta no Render para colocar as API

<https://dashboard.render.com/>

Hora de montar o FRONT END

Para criar o VITE, que será nosso conversor, iremos colocar o comando:

npm create vite

Depois dá um enter para confirmar instalação

Coloca o nome do projeto: frontend

Texto

Descrição gerada automaticamente

Escolher REACT e depois escolhe JAVA SCRIPT

Texto

Descrição gerada automaticamente

Entra na pasta, CD FRONTEND

Instalar o NPM i

Depois faz **npm run dev** para colocar o endereço local de trabalho, no caso foi o 127.0.0.1:5173

ALT + SHIFT + O apaga os imports que não estão sendo usado

Faz a criação de pastas

Relógio digital com números

Descrição gerada automaticamente com confiança baixa

Caso necessite de atalhos, pode instalar plugin no VSCODE

dsznajder.es7-react-js-snippets

Para pegar boas fontes:

Fonts.google.br

Para saber mais do DISPLAY FLEX

<https://origamid.com/projetos/flexbox-guia-completo/>

Para jogar com o display flex 27 niveis

<https://flexboxfroggy.com/>

Para importar de forma dinâmica na tela, no caso, no APP, precisa entender de ROTAS

Necessário usar a biblioteca <https://www.npmjs.com/package/react-router-dom>

Para instalar dá o comando **npm i react-router-dom** dentro da pasta FRONTEND

Texto

Descrição gerada automaticamente

Logo é instalado o pacote

  "scripts": {

    "dev": "vite",

    "build": "vite build",

    "preview": "vite preview"

  },

  "dependencies": {

    "react": "^18.2.0",

    "react-dom": "^18.2.0",

    "react-router-dom": "^6.4.3"

Para funcionar, é necessário importar o ROUTES e ROUTE no arquivo APP.JSX

import {Route, Routes} from "react-router-dom"

O react-router-dom precisa implementar outras coisas, indo no arquivo MAIN.JSX para mudar algumas coisas

import React from 'react'

import ReactDOM from 'react-dom/client'

import { BrowserRouter } from 'react-router-dom'

import App from './App'

import './index.css'

ReactDOM.createRoot(document.getElementById('root')).render(

  <*React.StrictMode*>

    <*BrowserRouter*>

      <*App* />

    </*BrowserRouter*>

  </*React.StrictMode*>

)

Envolve o APP com o BrowserRouter que automaticamente importante a biblioteca BrowserRouter from ‘react-router-dom’. Isso é muito doido

Caso tenha mudado a pasta para BACKEND no GITHUB ou para qualquer outra pasta devemos ir no RENDER.COM, entrar no DashBoard e mudar as Settings para ROOT Directory para o nome da pasta onde esta o arquivo JSON

Interface gráfica do usuário, Aplicativo, Teams

Descrição gerada automaticamente