1. Instalar as dependências para **backend**
   1. npm init -y
   2. npm i express dotenv mongoose
   3. npm i -D @types/express
   4. npm i -D ts-node ts-node-dev typescript
   5. tsc –init
2. Criar arquivo **.gitignore** adicionar no arquivo a pasta **node\_modules**
3. Criar o arquivo **.env | IMPORTANTE** Verificar nos requisitos qual a PORT que deve rodar!!!

PORT = 3010

JWT\_SECRET = @tokenJWT

DB\_URI = mongodb://127.0.0.1:27017/bdatividade

1. Adicionar o script no arquivo “package.json”

"scripts": {

"start": "ts-node ./src",

"dev": "ts-node-dev ./src"

},

1. Criar o arquivo index.ts ainda na raíz da pasta src
   1. Src/index.ts
   2. Adicionar o código abaixo para **subir o servidor**

import express from "express";

import cors from "cors";

import dotenv from "dotenv";

import routes from "./routes";

import { connect } from "./database/connection";

dotenv.config();

// será usado 3000 se a variável de ambiente não tiver sido definida

const PORT = process.env.PORT || 3000;

const app = express(); // cria o servidor e coloca na variável app

// suportar parâmetros JSON no body da requisição

app.use(express.json());

// configura o servidor para receber requisições de qualquer domínio

app.use(cors());

connect();

// inicializa o servidor na porta especificada

app.listen(PORT, () => {

  console.log(`Rodando na porta ${PORT}...`);

});

// define a rota para o pacote /routes

app.use(routes);

1. Utilize a estrutura abaixo de exemplo de pastas

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

1. crie dentro da pasta **database** o arquivo connection.ts e insira o código abaixo para conexão com o mongodb **IMPORTANTE |** lembre-se de ajustar o código para criar o banco corretamente, aqui estamos adicionando o nome do banco no **.env**

import mongoose from "mongoose";

import dotenv from "dotenv";

dotenv.config();

// A URI indica o IP, a porta e BD a ser conectado

const uri: string = process.env.DB\_URI || "";

// Salva o objeto mongoose em uma variável

const db = mongoose;

export function connect() {

  // Utiliza o método connect do Mongoose para estabelecer a conexão com

  // o MongoDB, usando a URI

  db.connect(uri, {

    serverSelectionTimeoutMS: 20000,

    maxPoolSize: 10,

  })

    .then(() => console.log("Conectado ao MongoDB"))

    .catch((e) => {

      console.error("Erro ao conectar ao MongoDB:", e.message);

    });

  // o sinal SIGINT é disparado ao encerrar a aplicação, geralmente, usando Crtl+C

  process.on("SIGINT", async () => {

    try {

      console.log("Conexão com o MongoDB fechada");

      await mongoose.connection.close();

      process.exit(0);

    } catch (error) {

      console.error("Erro ao fechar a conexão com o MongoDB:", error);

      process.exit(1);

    }

  });

}

export async function disconnect() {

  console.log("Conexão com o MongoDB encerrada");

  await db.disconnect();

}

1. agora você deve criar os models que irão criar as coleções dentro do bd no mongodb utilizando o mongoose, abaixo está um exemplo:

para que você consiga criar um código conforme o exemplo você pode enviar comandos para o chat GPT como o abaixo**"ChatGPT, você pode criar um arquivo TypeScript chamado category.ts com o seguinte código utilizando Mongoose? O schema deve ter a propriedade name com as seguintes características:**

* **Tipo: String**
* **Trim: true**
* **Lowercase: true**
* **Unique: true**
* **Required: [true, 'O nome é obrigatório']**

**Além disso, o schema deve ter uma configuração toJSON que transforma a saída JSON, adicionando uma propriedade id igual a \_id, e removendo as propriedades \_id e \_\_v. O modelo deve ser exportado com o nome category e a coleção deve ser chamada categories."**

import mongoose, { Schema } from "mongoose";

const categorySchema = new Schema({

    name:{

        type: String,

        trim: true,

        lowercase: true,

        unique: true,

        required: [true, "O nome é obrigatório"]

    }

},{

    toJSON: {

        transform: function(doc,ret,options){

            ret.id = ret.\_id;

            delete ret.\_id;

            delete ret.\_\_v;

        }

    }

});

export default mongoose.model("category", categorySchema, "categories");

,

1. Agora você deve criar os controllers, lembrando que o controller deve obter muitas vezes o CRUD | Create, Read, Update and Delete, sendo o Read o LIST