PRÁTICA 16

REQUISITOS

- Criar um programa que declare um array contendo 5 objetos representando países, sendo que cada país deve ter os seguintes atributos: nome e população.
- Persistir os objetos do array no banco de dados devweb2 no mongoDB na coleção países.
- No mesmo projeto, criar um servidor web Express com a rota /paises utilizando o verbo GET. Nessa rota deve ser passado o parâmetro populacao via query.
- Caso o parâmetro não seja informado ou não seja um número, a rota deve retornar um status 400 com uma mensagem orientativa.
- Caso o parâmetro seja informado, a rota deverá conectar ao mongoDB e recuperar todos os países que tem a população maior ou igual ao parâmetro informado. A resposta da rota deverá ser um array json de países.

RESOLUÇÃO

db/index.ts

```
// Importa Classe MongoClient
import { MongoClient } from "mongodb";

// String de Conexão
const uri = "mongodb://admin:admin@127.0.0.1:27017/devweb2";

// Retorna um conexão ativa com o MongoDB
const getMongoConn = async (): Promise<MongoClient> => {
    const client = new MongoClient(uri);
    const conn = await client.connect();
    return conn;
}
```

```
// Exporta a função de conexão
export default getMongoConn;
```

models/pais.ts

```
// Classe Modelo País
export default class Pais {
    nome: string;
    populacao: number;

    constructor(nome: string, populacao: number) {
        this.nome = nome;
        this.populacao = populacao;
    }

}
```

index.ts

```
// Importa a classe MongoClient
import { MongoClient } from "mongodb";
// Importa a função de conexão
import getMongoConn from "./db";
// Importa a classe País
import Pais from "./models/pais.js";

// Array com os países escolhidos
const paises: Pais[] = [
    new Pais("Brasil", 203062512),
    new Pais("Estados Unidos", 339987103),
    new Pais("Egito", 102334404),
    new Pais("Reino Unido", 67886011),
    new Pais("Argentina", 45195774)
];
```

```
// Função principal da aplicação
const main = async () => {
    // Variável que irá armazenar o objeto de conexão
    let conn: MongoClient | null = null;
    try {
        // Pega a conexão ativa
        conn = await getMongoConn();
        // Seleciona o banco de dados devweb2
        const db = conn.db();
        // Seleciona a coleção países
        const paisesCollection = db.collection("paises");
        // Deleta todos os países da coleção
        await paisesCollection.deleteMany({});
        // Insere cada país na coleção
        for (let pais of paises) {
            await paisesCollection.insertOne(pais);
        }
        // Busca os países inseridos e imprime na tela
        const docs = await paisesCollection.find().toArray();
        console.log(docs);
    } catch (err) {
        console.log((err as Error).message);
    } finally {
        conn?.close();
    }
}

// Chamada da função principal
main();
```

server.ts

```
// Importa a função express e as interfaces
import express, { Request, Response } from "express";
// Importa a função de conexão do MongoDB
import getMongoConn from "./db";
```

```
// Importa a classe MongoClient
import { MongoClient } from "mongodb";
// Importa a função CORS
import cors from "cors";

// Porta do servidor
const port = 3000;
// Objeto da aplicação
const app = express();

// Ativa o CORS
app.use(cors());

/**
 * Rota Países
 */
app.get("/paises", async (req: Request, res: Response) => {
    // Recuperar parâmetros query
    const { populacao } = req.query;
    // Variável de conexão
    let conn: MongoClient | null = null;
    try {
        // Valida se populacao foi informada
        if (typeof populacao !== "string") {
            throw new Error("Parâmetro populacao deve ser informado!");
        }
        // Valida se populacao é um número
        let populacaoNumber = parseInt(populacao);
        if (isNaN(populacaoNumber)) {
            throw new Error("Parâmetro populacao não é válido!");
        }

        try {
            // Retorna conexão ativa
            conn = await getMongoConn();
            // Seleciona o banco de dados devweb2
            const db = conn.db();
            // Seleciona a coleção países
            const paises = db.collection("paises");
```

```
            // Busca os países com populacao maior ou igual
            // Semelhante: where populacao >= ?
            const docs = await paises.find({
                populacao: { $gte: populacaoNumber }
            }).toArray();
            // Retorna array JSON
            res.status(200).json(docs);
        } catch (err) {
            // Erros do servidor
            res.status(500).json({
                message: (err as Error).message
            });
        } finally {
            conn?.close();
        }

    } catch (err) {
        // Erros do cliente
        res.status(400).json({
            message: (err as Error).message
        });
    }
});

// Inicia o servidor web
app.listen(port, () => {
    console.log(`Servidor sendo executado na porta ${port}`);
});
```