Descrição da atividade: codificar um servidor para listar dados de estações meteorológicas armazenadas no MongoDB.

Nota: 0,25 pts. na média final.

Data de entrega: na aula de 09/maio.

Forma de entrega: individual e presencial.

Objetivos:

- ORM Mongoose;
- Esquemas e modelos do Mongoose;
- Modelo de dados incorporados;
- Consultas usando os métodos find e aggregate do MongoDB;
- Node.js e Express.

Sobre os dados: Banco de Dados Meteorológicos do INMET – os dados foram baixados de https://portal.inmet.gov.br/dadoshistoricos, uma breve explicação se encontra em https://bdmep.inmet.gov.br. Foram baixados os dados das estações de coletas de dados meteorológicos dos estados do Espírito Santo, Rio de Janeiro e São Paulo do período de 01/01/2024 a 31/03/2024.

Passos para criar o projeto:

1. Crie uma pasta de nome servidor no local de seu interesse;
2. Execute os seguintes comandos, no terminal do VS Code, para configurar o projeto e instalar as dependências:

```
npm init -y
tsc --init
npm i express dotenv cors mongoose
npm i -D @types/express @type/cors ts-node ts-node-dev typescript
```

3. Execute os comandos a seguir para instalar as dependências necessárias para manipular arquivos CSV:

```
npm i fs-extra
npm i -D @types/fs-extra
```

4. Crie o arquivo `.env` na raiz do projeto e coloque nele a seguinte variável para subirmos na porta 3004:

```
PORT = 3004
```

5. Na propriedade `scripts`, do arquivo package.json, forneça os comandos para rodar a aplicação. O comando `load` será usado para carregar os arquivos CSV numa coleção do MongoDB.

```
"scripts": {
  "start": "ts-node ./src",
  "dev": "ts-node-dev ./src",
```

```
    "load": "ts-node ./src/controllers/readCSV"
  },
```

6. Descompactar os arquivos CSV na pasta `dados` do projeto. O correto é ter 78 arquivos CSV na pasta `dados`;
7. Coloque o código a seguir no arquivo src/models/connection.ts. Esse faz a conexão com o `bdmeteorologico` no MongoDB. O `bdmeteorologico` será criado pelo mongoose se ele não existir;

src/models/connection.ts

```
import mongoose from "mongoose";

// A URI indica o IP, a porta e BD a ser conectado
const uri = "mongodb://127.0.0.1:27017/bdmeteorologico";

// 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: 12000,
    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();
}
```

8. Coloque o código a seguir no arquivo src/models/Estacao.ts. Esse código é usado para definir os esquemas e modelos das coleções no mongoose. Os esquemas e modelos serão utilizados apenas pelo

ORM mongoose, visto que eles não serão enviados para o MongoDB. Foi utilizado o modelo de dados incorporados, onde as leituras são documentos aninhados no array do campo `leituras`;

src/models/Estacao.ts

```
import mongoose from "mongoose";
const { Schema } = mongoose;

const LeituraSchema = new Schema({
    datahora: Date,
    precipitacao: Number,
    pressaoAtmNivel: Number,
    pressaoAtmMax: Number,
    pressaoAtmMin: Number,
    radiacao: Number,
    temperaturaAr: Number,
    temperaturaOrvalho: Number,
    temperaturaMax: Number,
    temperaturaMin: Number,
    temperaturaOrvalhoMax: Number,
    temperaturaOrvalhoMin: Number,
    umidadeRelativaMax: Number,
    umidadeRelativaMin: Number,
    umidadeRelativa: Number,
    ventoDirecao: Number,
    ventoRajada: Number,
    ventoVelocidade: Number,
});

// define os schemas
const EstacaoSchema = new Schema({
  regiao: String,
  uf: String,
  estacao: String,
  codigo: String,
  latitude: Number,
  longitude: Number,
  altitude: Number,
  dataFundacao: Date,
  leituras: [LeituraSchema],
});

// mongoose.model compila o modelo
const LeituraModel = mongoose.model("Leitura", LeituraSchema);
const EstacaoModel = mongoose.model("Estacao", EstacaoSchema, "estacoes");

export { LeituraModel, EstacaoModel };
```

9. Coloque o código a seguir no arquivo src/types/index.ts. Por enquanto existem apenas as definições de tipos para `Estacao` e `Leitura`;

src/types/index.ts

```
export interface Estacao {
  regiao: string|undefined;
  uf: string|undefined;
  estacao: string|undefined;
  codigo: string|undefined;
  latitude: number|undefined;
  longitude: number|undefined;
  altitude: number|undefined;
  dataFundacao: Date|undefined;
  leituras: Leitura[];
}

export interface Leitura {
    datahora: Date|undefined;
    precipitacao: number|undefined;
    pressaoAtmNivel: number|undefined;
    pressaoAtmMax: number|undefined;
    pressaoAtmMin: number|undefined;
    radiacao: number|undefined;
    temperaturaAr: number|undefined;
    temperaturaOrvalho: number|undefined;
    temperaturaMax: number|undefined;
    temperaturaMin: number|undefined;
    temperaturaOrvalhoMax: number|undefined;
    temperaturaOrvalhoMin: number|undefined;
    umidadeRelativaMax: number|undefined;
    umidadeRelativaMin: number|undefined;
    umidadeRelativa: number|undefined;
    ventoDirecao: number|undefined;
    ventoRajada: number|undefined;
    ventoVelocidade: number|undefined;
}
```

10. Coloque o código a seguir no arquivo src/controllers/EstacaoController.ts. Por enquanto esse código possui apenas o método `insert` para inserir 1 documento na coleção `estacoes` do MongoDB;

src/controllers/EstacaoController.ts

```
import { Request, Response } from "express";
import { EstacaoModel } from "../models/Estacao";
import { Estacao } from "../types";

class EstacaoController {
  // Insere um documento na coleção estacoes
  public async insert(estacao: Estacao): Promise<void> {
```

```
    try {
      const document = new EstacaoModel(estacao);
      await document.save(); // insere na coleção
    } catch (error: any) {
      console.log(estacao.estacao, error.message);
    }
  }
}

export default new EstacaoController();
```

11. Coloque o código a seguir no arquivo src/controllers/readCSV.ts. Esse código faz a leitura dos arquivos CSV e carregam nas coleções do `bdmeteorologico` no MongoDB. O `bdmeteorologico` será criado pelo mongoose se ele não existir;

src/controllers/readCSV.ts

```
import fs from "fs-extra";
import { Estacao } from "../types";
import { connect, disconnect } from "../models/connection";
import controller from "./EstacaoController";

// Pasta onde estão os arquivos CSV
const pasta = "./dados";

function getDataFundacao(data: string) {
  try {
    const temp = data.split("/");
    return new Date(
      Date.UTC(
        parseInt("20" + temp[2]),
        parseInt(temp[1]) - 1,
        parseInt(temp[0])
      )
    );
  } catch (e: any) {
    return undefined;
  }
}

function getDataHorario(data: string, hora: string): Date | undefined {
  try {
    const temp = data.split("/");
    const h = parseInt(hora.substring(0, 2));
    const m = parseInt(hora.substring(2, 4));
    return new Date(
      Date.UTC(
        parseInt(temp[0]),
        parseInt(temp[1]) - 1,
```

```
        parseInt(temp[2]),
        h,
        m,
        0
      )
    );
  } catch (e: any) {
    return undefined;
  }
}

function getValue(input: string): number | undefined {
  const value = parseFloat(input.replace(",", "."));
  return isNaN(value) ? undefined : value;
}

// Função para ler arquivos CSV na pasta
async function lerArquivosCSV(): Promise<void> {
  // Objeto usado para manter os dados de um arquivo CSV
  let estacao: Estacao;

  try {
    // Obtém uma lista de todos os arquivos na pasta
    const files = await fs.readdir(pasta);
    let count = 1;
    // Para cada arquivo na pasta
    for (const file of files) {
      estacao = {} as Estacao;
      try {
        // Verifica se é um arquivo CSV
        if (file.endsWith(".CSV")) {
          // Caminho completo do arquivo
          const filePath = `${pasta}/${file}`;
          // Lê o conteúdo do arquivo CSV
          const fileContent = await fs.readFile(filePath, "utf8");
          // Divide o conteúdo do arquivo em linhas
          const linhasCSV = fileContent.split("\n");
          // Parse do conteúdo CSV linha por linha
          for (let i = 0, linha; i < linhasCSV.length; i++) {
            linha = linhasCSV[i].split(";");
            if (linha.length >= 2 && linha[0] === "REGIAO:") {
              estacao.regiao = linha[1];
            } else if (linha.length >= 2 && linha[0] === "UF:") {
              estacao.uf = linha[1];
            } else if (linha.length >= 2 && linha[0] === "ESTACAO:") {
              estacao.estacao = linha[1];
            } else if (linha.length >= 2 && linha[0] === "CODIGO (WMO):") {
              estacao.codigo = linha[1];
            } else if (linha.length >= 2 && linha[0] === "LATITUDE:") {
```

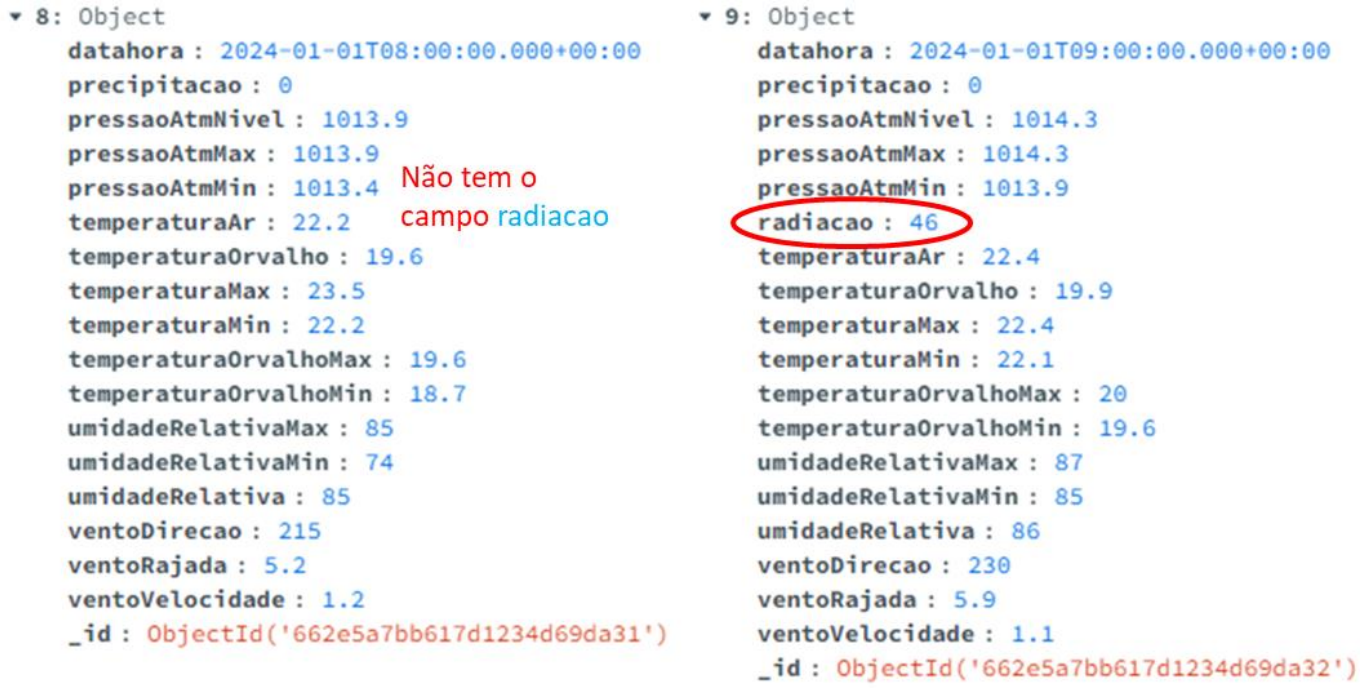
```
            estacao.latitude = getValue(linha[1]);
          } else if (linha.length >= 2 && linha[0] === "LONGITUDE:") {
            estacao.longitude = getValue(linha[1]);
          } else if (linha.length >= 2 && linha[0] === "ALTITUDE:") {
            estacao.altitude = getValue(linha[1]);
          } else if (linha.length >= 2 && linha[0] === "DATA DE FUNDACAO:") {
            estacao.dataFundacao = getDataFundacao(linha[1]);
          } else if (linha.length >= 20 && !linha[0].startsWith("Data")) {
            if( !estacao.leituras ){
              estacao.leituras = [];
            }
            estacao.leituras.push({
              datahora: getDataHorario(linha[0], linha[1]),
              precipitacao: getValue(linha[2]),
              pressaoAtmNivel: getValue(linha[3]),
              pressaoAtmMax: getValue(linha[4]),
              pressaoAtmMin: getValue(linha[5]),
              radiacao: getValue(linha[6]),
              temperaturaAr: getValue(linha[7]),
              temperaturaOrvalho: getValue(linha[8]),
              temperaturaMax: getValue(linha[9]),
              temperaturaMin: getValue(linha[10]),
              temperaturaOrvalhoMax: getValue(linha[11]),
              temperaturaOrvalhoMin: getValue(linha[12]),
              umidadeRelativaMax: getValue(linha[13]),
              umidadeRelativaMin: getValue(linha[14]),
              umidadeRelativa: getValue(linha[15]),
              ventoDirecao: getValue(linha[16]),
              ventoRajada: getValue(linha[17]),
              ventoVelocidade: getValue(linha[18]),
            });
          }
        }

        // cada arquivo será um documento na coleção
        await controller.insert(estacao);
        console.log(`${count++} - ${estacao.estacao}`);
      }
    } catch (e: any) {
      console.log(`Erro no arquivo ${file}`);
    }
  }
} catch (err: any) {
  console.error("Erro ao ler a pasta:", err.message);
}
}

// conecta ao MongoDB antes de escrever
connect();
```

```
// Chama a função para ler os arquivos CSV
lerArquivosCSV().finally(async () => await disconnect());
```

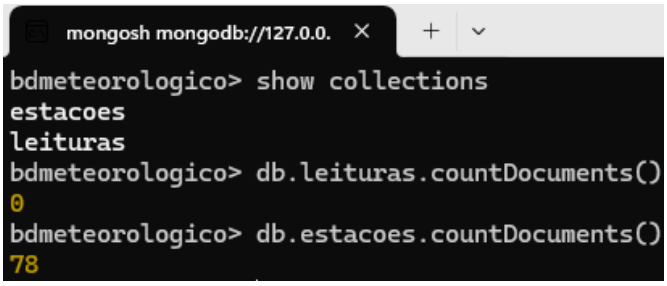
12. Execute o comando `npm run load` no terminal do VS Code para ler os arquivos CSV e carregar na coleção `estacoes` do `bdmeteorologico`;

```
PS D:\Atividade - MongoDB\servidor> npm run load

> servidor@1.0.0 load
> ts-node ./src/controllers/readCSV

Conectado ao MongoDB
1 - VITORIA
2 - SANTA TERESA
3 - LINHARES
4 - ALFREDO CHAVES
76 - CACHOEIRA PAULISTA
77 - SAO SIMAO
78 - SAO PAULO - INTERLAGOS
Conexão com o MongoDB encerrada
```

Por ser um modelo de dados flexível cada documento pode ter campos distintos. Na figura a seguir tem-se os documentos que estão nas posições 8 e 9 do array do campo `leituras` do documento que possui `estacao:"VITORIA"`. Veja que o campo `radiacao` não existe em ambos os documentos.



13. No shell do MongoDB Shell (mongosh) verifique se foram criadas as coleções `leituras` e `estacoes`. A coleção `leituras` foi criada ao definir o esquema e o modelo no arquivo src/models/Estacao.ts. Porém, ela não receberá documentos, visto que as leituras serão subdocumentos do campo `leituras` da coleção `estacoes`;



14. No arquivo src/index.ts defina o código para subir o servidor. Observe que:
    - A chamada da função `connect()` criará um pool de conexão para o `bdmeteorologico` do MongoDB;
    - Você precisará codificar as rotas na pasta `routes`.

src/index.ts

```
import express from "express";
import cors from "cors";
import dotenv from "dotenv";
import routes from "./routes";
import { connect } from "./models/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());

// conecta ao MongoDB no início da aplicação
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);
```

Exercícios

a) Codificar a rota HTTP GET `/estacao/lista` para retornar os campos `uf`, `estacao`, `latitude` e `longitude` de todas as estações. Apresente o resultado ordenado pelo campo `estacao`.

O resultado será um array com 78 objetos.

Observação: o comando para fazer a consulta no BD deverá estar na classe `EstacaoController`.

Dicas:

- Use o método `find(query, projection, options);`
- No parâmetro `projection` forneça os campos `uf`, `estacao`, `latitude` e `longitude`, e retire o campo obrigatório `_id`;
- Forneça a propriedade `sort` no objeto do parâmetro `options`.

localhost:3004/estacao/lista


```
[
  {
    "uf": "ES",
    "estacao": "AFONSO CLAUDIO",
    "latitude": -20.10416666,
    "longitude": -41.10694444
  },
  {
    "uf": "ES",
    "estacao": "ALEGRE",
    "latitude": -20.75055555,
    "longitude": -41.48888888
  },
  {
    "uf": "ES",
    "estacao": "ALFREDO CHAVES",
    "latitude": -20.63638888,
    "longitude": -40.74194444
  },
```


b) Codificar a rota HTTP GET `/estacao/leiturasporestacao` para retornar a quantidade de leituras por estação. Apresente o resultado ordenado pelo campo `estacao`.

O resultado será um array com 78 objetos.

Observação: o comando para fazer a consulta no BD deverá estar na classe `EstacaoController`.

Dicas:

- Use o método `aggregate([{$project}, {$sort}])` com os estágios `$project` e `$sort`. Cada estágio precisa estar em um objeto, ou seja, delimitados por chaves;
- No estágio `$project`, use o operador `$size` para obter a quantidade de elementos do array que está no campo `leituras`;
- No estágio `$sort`, use o campo `estacao` para ordenar o resultado.

Para mais detalhes sobre o operador $size acesse https://www.mongodb.com/docs/manual/reference/operator/aggregation/size.

localhost:3004/estacao/leiturasporestacao


```
[
  {
    "estacao": "AFONSO CLAUDIO",
    "quantidade": 2184
  },
  {
    "estacao": "ALEGRE",
    "quantidade": 2184
  },
  {
    "estacao": "ALFREDO CHAVES",
    "quantidade": 2184
  },
  {
    "estacao": "ANGRA DOS REIS",
    "quantidade": 2184
  },
  {
    "estacao": "ARIRANHA",
    "quantidade": 2184
  },
```

c) Codificar a rota HTTP GET `/estacao/estatisticatemperatura` que recebe como parâmetro na URL o nome de uma estação e retorna a média, a mínima e a máxima temperatura do ar calculada no campo `temperaturaAr` de cada leitura da estação.

O resultado será um array com 1 objeto.

Observação: o comando para fazer a consulta no BD deverá estar na classe `EstacaoController`.


```
[{"media":27.426597582037996,"minima":21.5,"maxima":37.9,"estacao":"VITORIA"}]
```

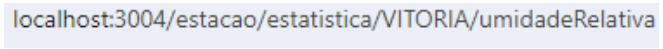

Dicas:

- Use o método `aggregate([{$match},{$unwind},{$group},{$project}]);`
- No estágio `$match` use a condição `estacao:NomeDaEstacaoPassadaComoParâmetro`;
- Forneça o campo `"$leituras"` para o estágio `$unwind`. Para calcular a média do campo `temperaturaAr` dos subdocumentos que estão no campo `leituras`, foi necessário usar o operador `$unwind` para "desconstruir" o array `leituras` em documentos individuais antes de calcular a média;
- No estágio `$group`,
  - Será necessário agrupar pelo campo `estacao`;
  - Use os operadores `$avg`, `$min` e `$max` para obter, respectivamente, a média, o mínimo e o máximo no campo `"$leituras.temperaturaAr"`.

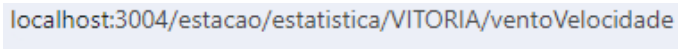
d) Codificar a rota HTTP GET `/estacao/estatistica` que recebe como parâmetro na URL o nome de uma estação e o nome de um campo da coleção e retorna o valor médio, o valor mínimo e o valor máximo calculado no campo fornecido como parâmetro.

O resultado será um array com 1 objeto.

Observação: o comando para fazer a consulta no BD deverá estar na classe `EstacaoController`.




```
[
  {
    "media": 73.6976331360947,
    "minima": 29,
    "maxima": 100,
    "estacao": "VITORIA",
    "parametro": "umidadeRelativa"
  }
]
```


```
[
  {
    "media": 1.4959630911188,
    "minima": 0.1,
    "maxima": 4.8,
    "estacao": "VITORIA",
    "parametro": "ventoVelocidade"
  }
]
```


Dica: o nome do campo precisa ser um dos campos definidos no esquema `LeituraSchema` do arquivo `src/models/estacao.ts`.

e) Codificar a rota HTTP GET `/estacao/leituras` que recebe como parâmetro na URL o nome de uma estação, o nome de um campo da coleção, a data de início e a data de fim, e retorna todos os valores desse sensor no intervalo de datas fornecido.

O resultado será um array com 1 objeto e dentro desse objeto deverá ter a propriedades leituras e estacao. A propriedade leituras terá um array com 48 registros.

Observação: o comando para fazer a consulta no BD deverá estar na classe `EstacaoController`.

Dicas:

- Use o método `aggregate([{$unwind},{$match},{$group},{$project}]);`
- No estágio `$unwind` forneça o campo `"$leituras"` para descontruir o array de leituras;
- No estágio `$match` use a condição

```
leituras.datahora:{
    $gte: new Date("yyyy-mm-dd T00:00:00.000Z"),
    $lte: new Date("yyyy-mm-dd T23:00:00.000Z")
}
```

- No estágio `$group`,
  - Será necessário agrupar pelo campo `estacao`;
  - Crie o campo `leituras` com o operador `$push` para concatenar as respostas em um array. Esse recurso criará a série temporal de `leituras` com as propriedades `leitura` e `data`: O termo `parâmetro` terá de ser substituído pelo nome de um campo da coleção leituras do BD:

```
leituras: {
  $push: {
    leitura: "$leituras.parâmetro",
    data: "$leituras.datahora",
  },
},
```

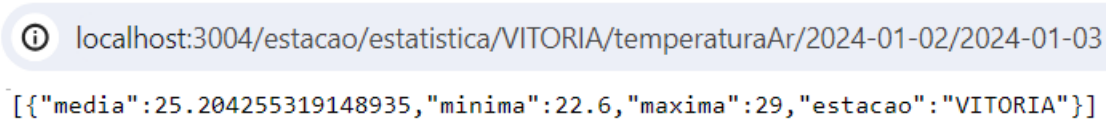
localhost:3004/estacao/leituras/VITORIA/temperaturaAr/2024-01-02/2024-01-03


```
[
  {
    "leituras": [
      {
        "leitura": 23,
        "data": "2024-01-02T00:00:00.000Z"
      },
      {
        "leitura": 23.2,
        "data": "2024-01-02T01:00:00.000Z"
      },
      ...
      {
        "leitura": 25.9,
        "data": "2024-01-03T22:00:00.000Z"
      },
      {
        "leitura": 26.1,
        "data": "2024-01-03T23:00:00.000Z"
      }
    ],
    "estacao": "VITORIA"
  }
]
```

f) Codificar a rota HTTP GET `/estacao/estatistica` que recebe como parâmetro na URL o nome de uma estação, o nome de um campo da coleção, a data de início e a data de fim, e retorna o valor médio, mínimo e máximo calculado no campo fornecido como parâmetro.

O resultado será um array com 1 objeto.

Observação: o comando para fazer a consulta no BD deverá estar na classe `EstacaoController`.



Dica:

- Use como base o comando do Exercício (e). A diferença é que no estágio `$group` podemos calcular as estatísticas diretamente usando os operadores `$avg`, `$min` e `$max`.

g) Codificar a rota HTTP GET `/estacao/intervalo` que recebe como parâmetro na URL o nome de uma estação, a data de início e a data de fim, e retorna todos as leituras no intervalo de datas fornecido.

O resultado será um array com 1 objeto e dentro desse objeto deverá ter a propriedades `leituras` e `estacao`. A propriedade `leituras` terá um array com 48 registros.

Observação: o comando para fazer a consulta no BD deverá estar na classe `EstacaoController`.

Dicas:

- Use o método `aggregate([{$unwind},{$match},{$group},{$project}]);`
- No estágio `$unwind` forneça o campo `"$leituras"` para descontruir o array de leituras;
- No estágio `$match` use a condição

```
leituras.datahora:{
    $gte: new Date("yyyy-mm-ddT00:00:00.000Z"),
    $lte: new Date("yyyy-mm-ddT23:00:00.000Z")
}
```

- No estágio `$group`,
  - Será necessário agrupar pelo campo `estacao`;
  - Crie o campo `leituras` com o operador `$push` para concatenar as respostas em um array. Esse recurso criará a série temporal de `leituras` com as propriedades `leitura` e `data`: O termo `parâmetro` terá de ser substituído pelo nome de um campo da coleção leituras do BD:

```
leituras: {
  $push: {
    leitura: "$leituras.parâmetro",
    data: "$leituras.datahora",
  },
},
```

localhost:3004/estacao/intervalo/VITORIA/2024-01-02/2024-01-03


```
[
  {
    "leituras": [
      {
        "data": "2024-01-02T00:00:00.000Z",
        "precipitacao": 0,
        "temperaturaAr": 23,
        "umidadeRelativa": 88,
        "ventoDirecao": 193,
        "ventoVelocidade": 1.1
      },
      {
        "data": "2024-01-02T01:00:00.000Z",
        "precipitacao": 0,
        "temperaturaAr": 23.2,
        "umidadeRelativa": 87,
        "ventoDirecao": 167,
        "ventoVelocidade": 0.6
      },
      ...
      {
        "data": "2024-01-03T23:00:00.000Z",
        "precipitacao": 0,
        "temperaturaAr": 26.1,
        "umidadeRelativa": 85,
        "ventoDirecao": 357,
        "ventoVelocidade": 0.7
      }
    ],
    "estacao": "VITORIA"
  }
]
```