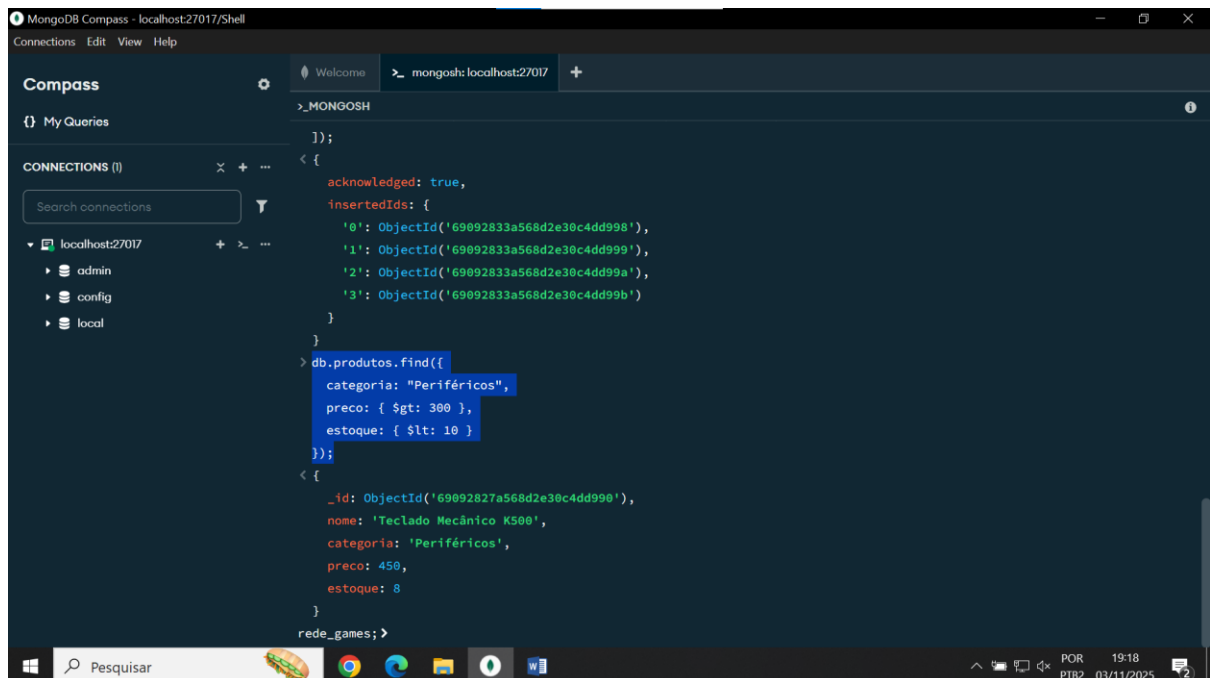


Atividade de Banco de Dados Não Relacional

Nome: Gustavo Henrique Ferreira Hammes

Questão 1 – Consulta com Operadores Lógicos Liste os produtos da categoria "Periféricos" cujo preço seja superior a R\$ 300 e estoque inferior a 10 unidades.

```
db.produtos.find({
  categoria: "Periféricos",
  preco: { $gt: 300 },
  estoque: { $lt: 10 }
});
```

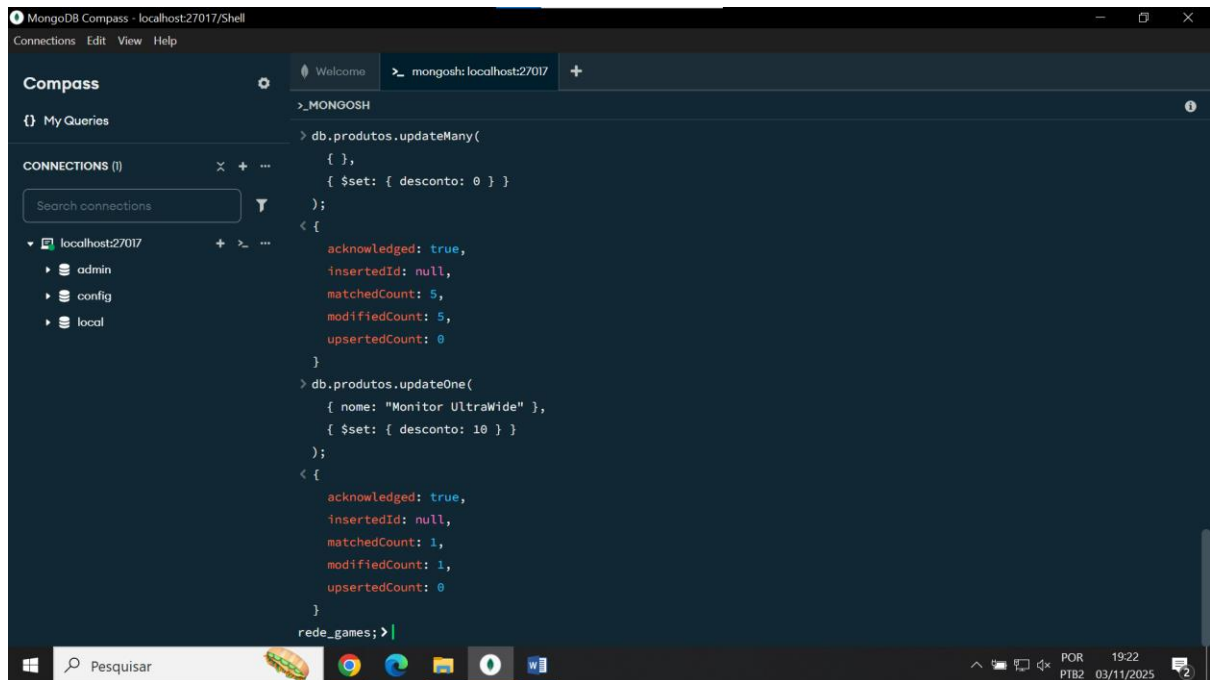


Questão 2 – Inserção com Operadores de Atualização Adicione o campo "desconto": 0 em todos os produtos e, depois, atualize o "Monitor UltraWide" para aplicar desconto de 10%

```
db.produtos.updateMany(
  {},
  { $set: { desconto: 0 } }
);
```

```
db.produtos.updateOne(  
  { nome: "Monitor UltraWide" },  
  { $set: { desconto: 10 } }  
);
```

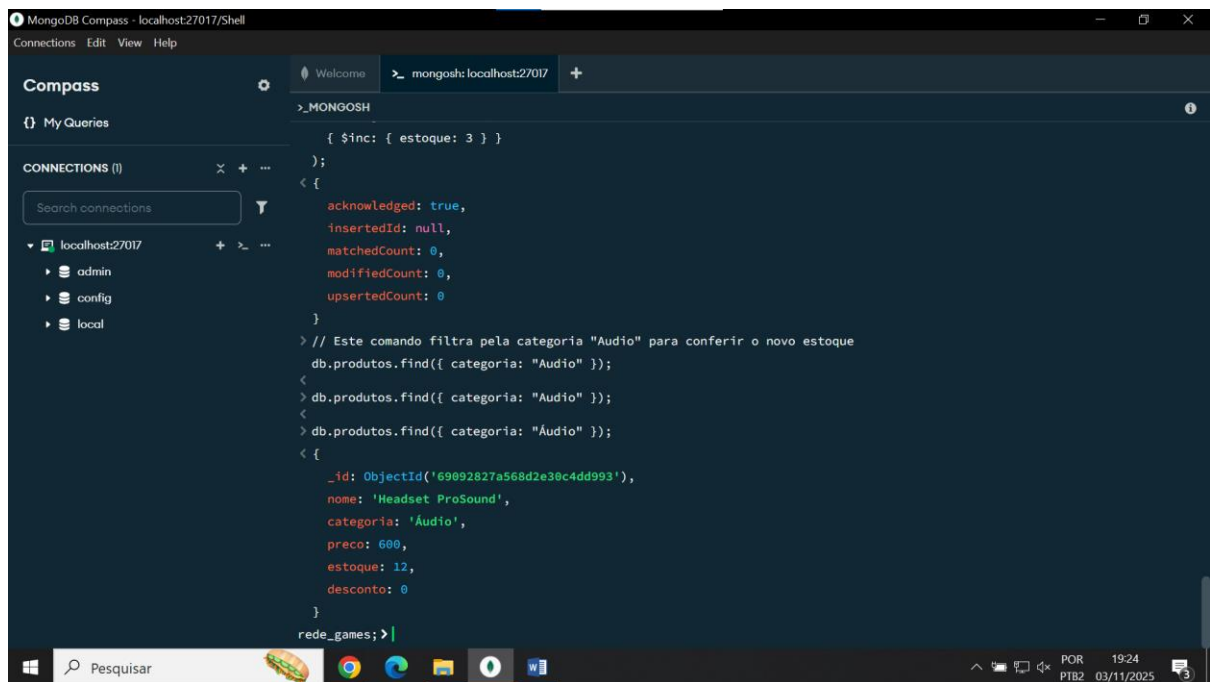
```
db.produtos.find();
```



Questão 3 – Incremento de Valores Aumente o estoque de todos os produtos da categoria “Áudio” em 3 unidades.

```
db.produtos.updateMany(  
  { categoria: "Audio" },  
  { $inc: { estoque: 3 } }  
);
```

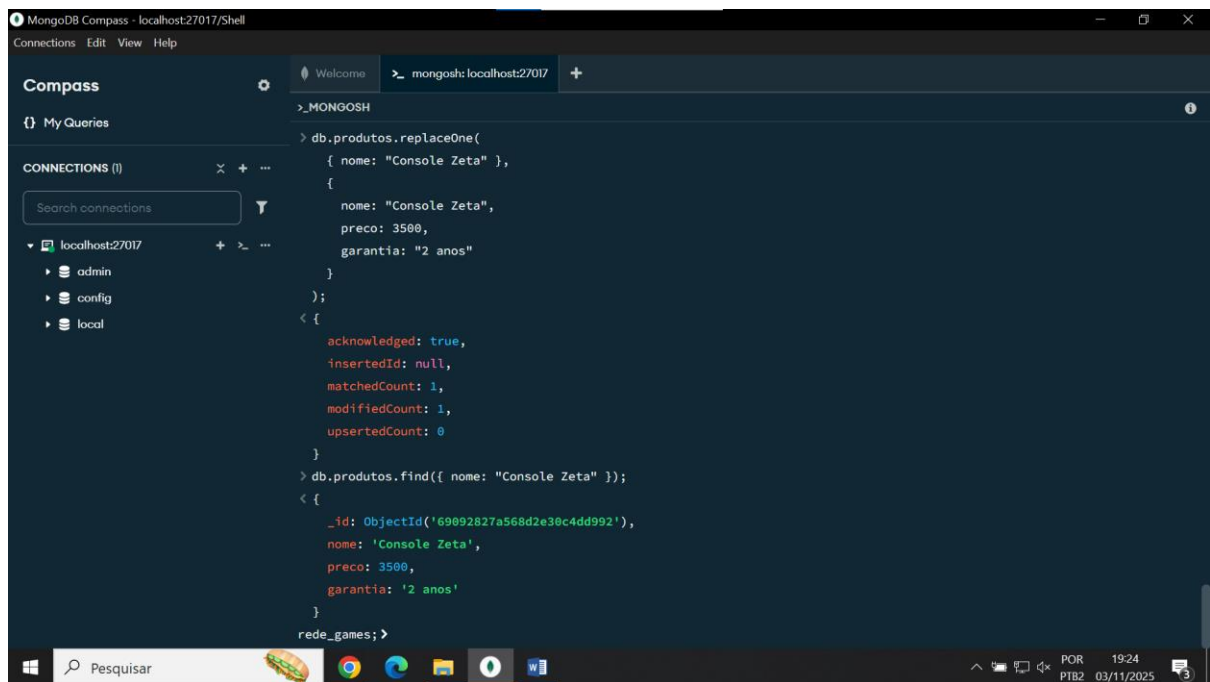
```
db.produtos.find({ categoria: "Audio" });
```



Questão 4 – Atualização e Substituição Substitua completamente o documento do produto “Console Zeta” por uma nova versão, mantendo o mesmo nome, mas com preço atualizado para 3500 e campo novo “garantia”: “2 anos”.

```
db.produtos.replaceOne(  
  { nome: "Console Zeta" },  
  {  
    nome: "Console Zeta",  
    preco: 3500,  
    garantia: "2 anos"  
  }  
);
```

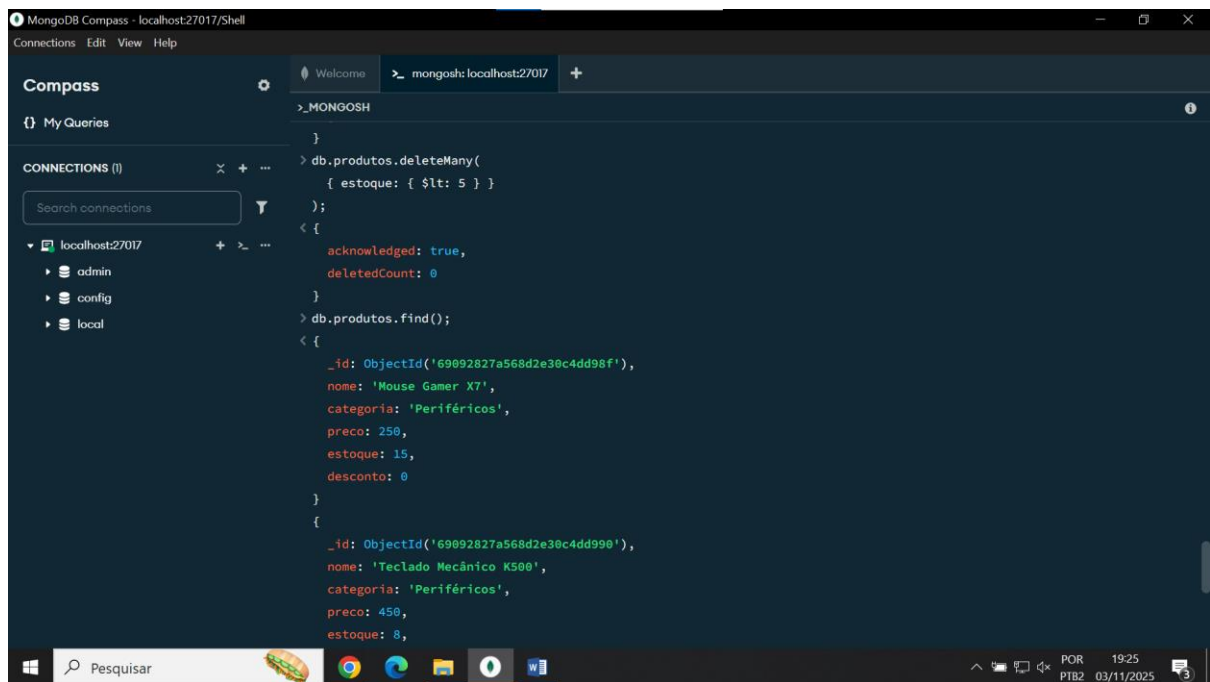
```
db.produtos.find({ nome: "Console Zeta" });
```



Questão 5 – Remoção Condicional Remova todos os produtos cujo estoque seja menor que 5

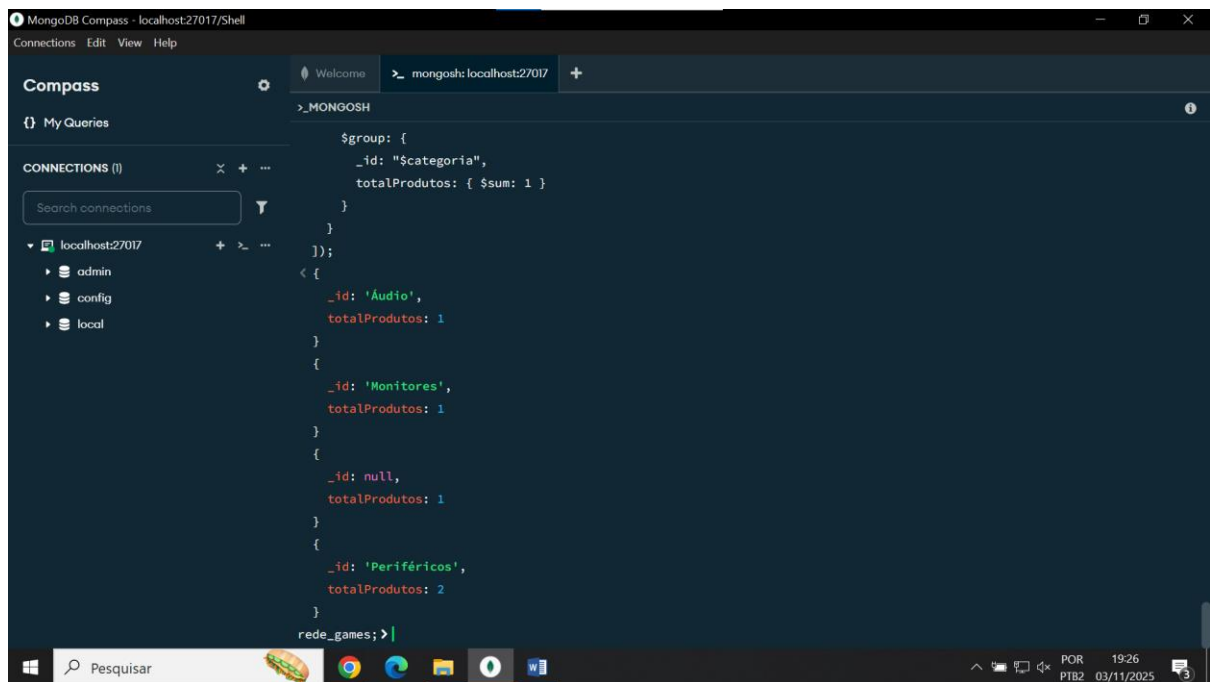
```
db.produtos.deleteMany(  
  { estoque: { $lt: 5 } }  
);
```

```
db.produtos.find();
```



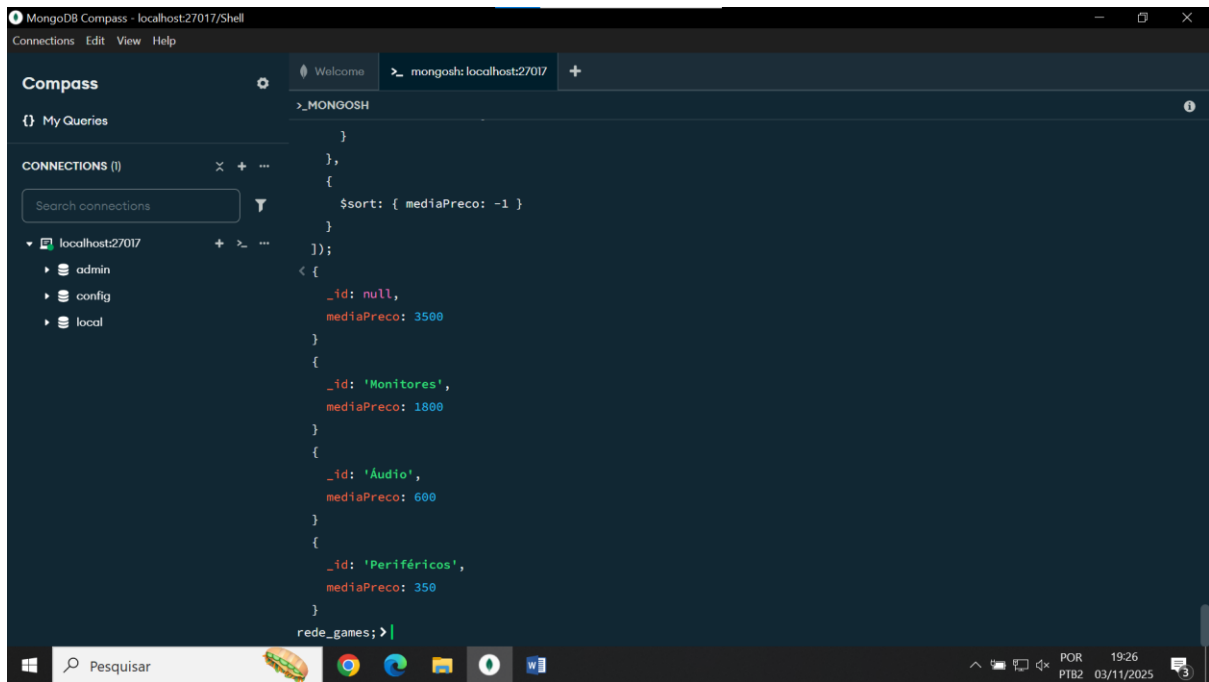
Questão 6 – Agregação e Contagem Conte quantos produtos existem em cada categoria.

```
db.produtos.aggregate([  
  
  {  
  
    $group: {  
  
      _id: "$categoria",  
  
      totalProdutos: { $sum: 1 }  
  
    }  
  
  }  
  
]);
```



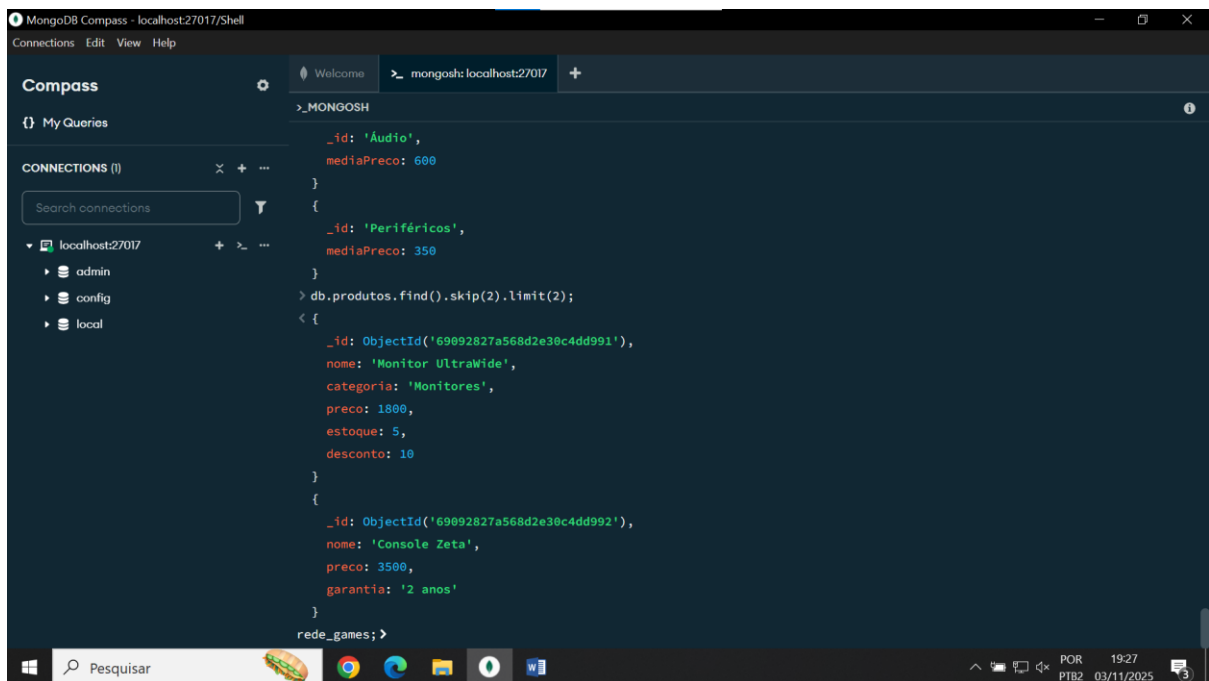
Questão 7 – Média de Preços por Categoria Calcule o preço médio dos produtos de cada categoria e ordene do maior para o menor.

```
db.produtos.aggregate([
{
  $group: {
    _id: "$categoria",
    mediaPreco: { $avg: "$preco" }
  }
},
{
  $sort: { mediaPreco: -1 }
}
]);
```



Questão 8 – Paginação Liste os produtos exibindo apenas 2 resultados por página, mostrando a segunda página.

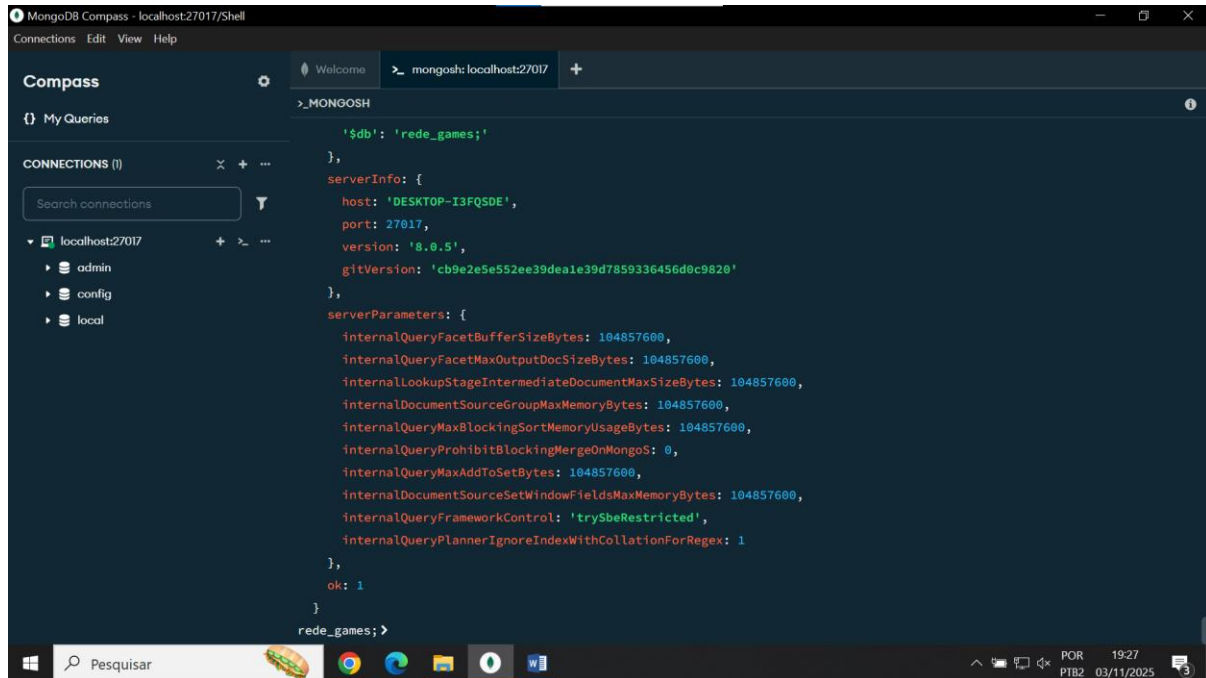
`db.produtos.find().skip(2).limit(2);`



Questão 9 – Indexação e Otimização Crie um índice no campo "categoria" e, em seguida, verifique se está sendo utilizado nas consultas.

```
db.produtos.createIndex({ categoria: 1 });
```

```
db.produtos.find(  
  { categoria: "Periféricos" }  
)>.explain("executionStats");
```



Questão 10 – Integração com API Crie uma rota em Node.js/Express que exiba todos os produtos com apenas nome e preço, limitando a 5 resultados

```
import express, { Request, Response } from 'express';  
import { MongoClient, Collection } from 'mongodb';  
  
// --- Configuração ---  
const app = express();  
const port = 3000;  
  
const url = 'mongodb://localhost:27017'; // URL de conexão  
const dbName = 'rede_games'; // Nome do banco de dados  
  
// --- Interface para tipar nosso resultado ---  
// Define que o objeto terá APENAS nome e preço  
interface ProdutoProjetado {  
  nome: string;  
  preco: number;  
}
```

```

// --- Função Principal de Conexão ---
async function main() {
  const client = new MongoClient(url);

  try {
    // Conecta ao MongoDB
    await client.connect();
    console.log('Conectado ao servidor MongoDB');

    const db = client.db(dbName);
    const collection: Collection = db.collection('produtos');

    // --- Definição da Rota da API ---
    app.get('/produtos', async (req: Request, res: Response) => {
      try {
        // Usamos a tipagem <ProdutoProjetado> para o .project()
        const produtos = await collection
          .find()
          .project<ProdutoProjetado>({ nome: 1, preco: 1, _id: 0 }) // 2.
          .limit(5) // 3. Limita a 5 resultados
          .toArray();

        res.json(produtos);

      } catch (err) {
        // Tratamento de erro para a rota
        if (err instanceof Error) {
          res.status(500).send({ error: 'Falha ao buscar produtos', details:
err.message });
        } else {
          res.status(500).send({ error: 'Ocorreu um erro desconhecido.' });
        }
      }
    });

    // Inicia o servidor Express
    app.listen(port, () => {
      console.log(`API da RedeGames (TS) rodando em
http://localhost:\${port}/produtos`);
    });

  } catch (err) {
    // Tratamento de erro da conexão principal
    console.error('Falha ao conectar ao MongoDB', err);
  }
}

```

```
    process.exit(1); // Encerra a aplicação se não conseguir conectar
  }
}

// Executa a função principal
main();
```

