Tarefa 4

Database 2

Gustavo Henrique de Jesus da Silva

Sumario

1. Introdução

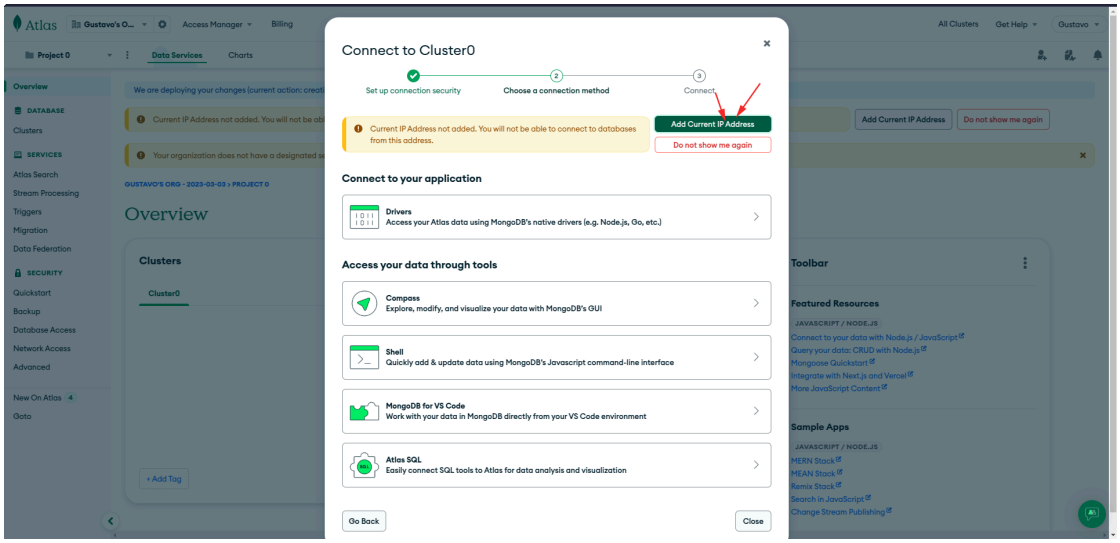
Este relatório documenta a implementação de um sistema **CRUD (Create, Read, Update, Delete)** utilizando **Java** e **MongoDB Atlas** (banco de dados em nuvem). O objetivo foi criar uma aplicação que interage com um cluster remoto, realizando operações básicas e avançadas, como agregações e indexação.

2. Configuração do Ambiente

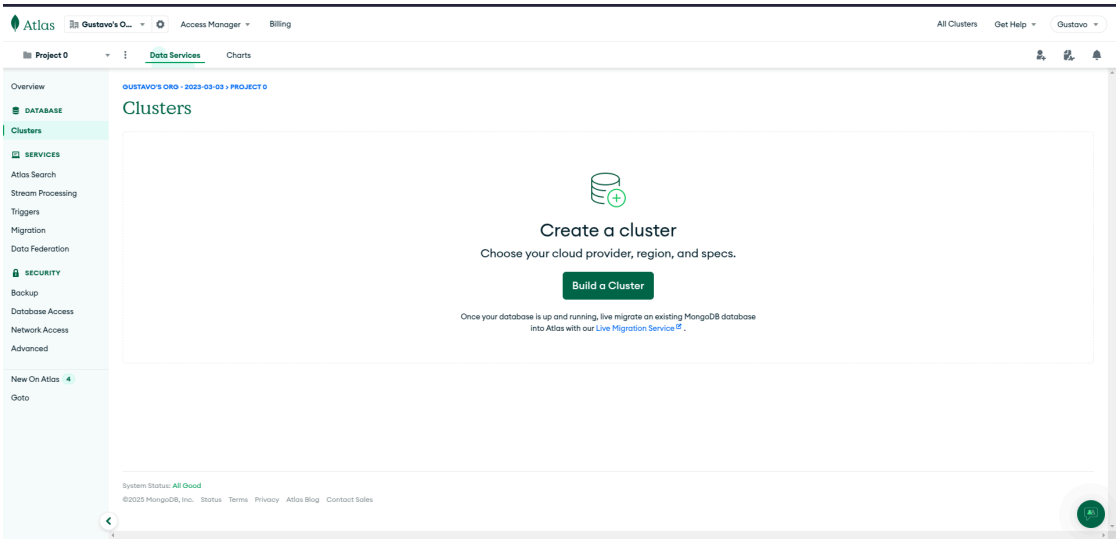
2.1 MongoDB Atlas

1. **Criação do Cluster Gratuito**

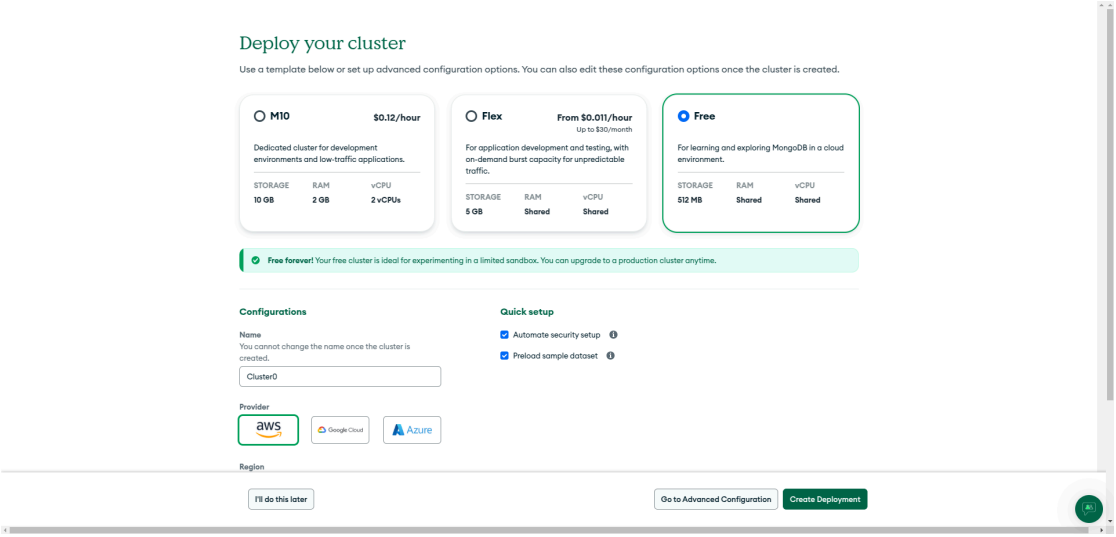
- Acessei o MongoDB Atlas (https://www.mongodb.com/atlas) e criei um cluster na camada gratuita (M0).
- Configurei um usuário com permissões de leitura e escrita.
- Adicionei meu endereço IP à lista de permissões.



add o endereço ip



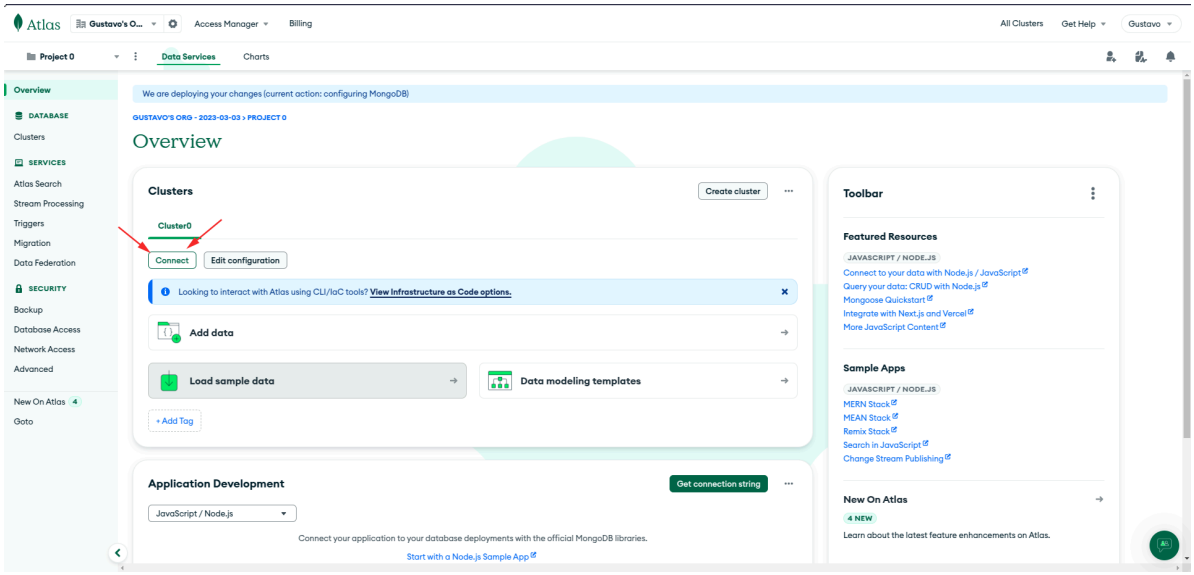
criando um cluster



configurando cluster

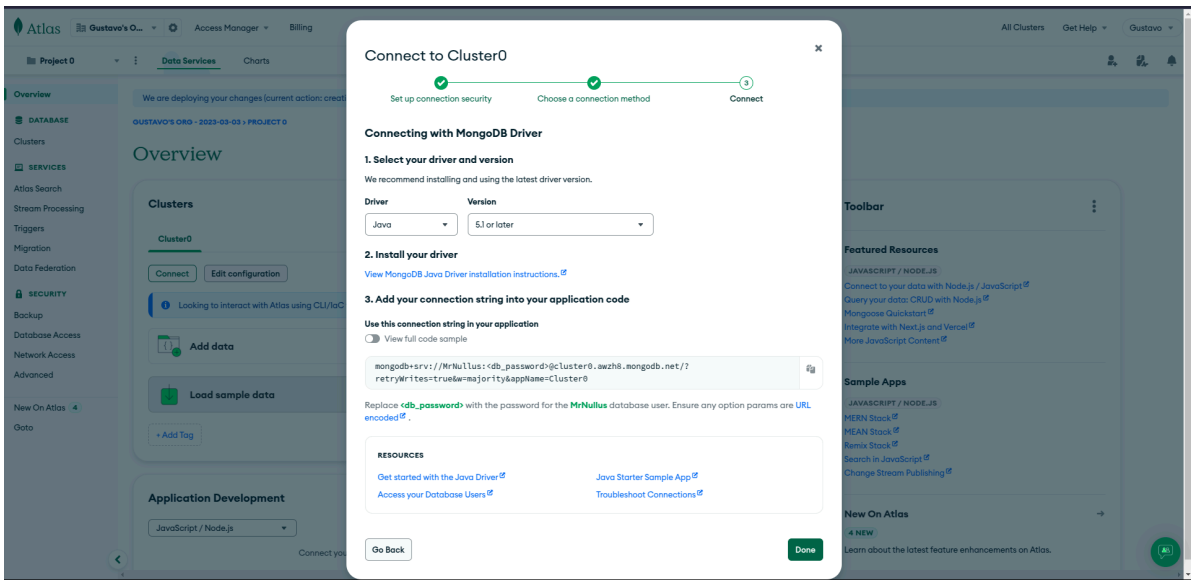
2. **Obtenção da String de Conexão**

- Na seção "**Connect**", selecionei "**Connect your application**".



1742836104644.png

- Copiei a **URI de conexão**, substituindo `<password>` e `<dbname>` pelas minhas credenciais.



1742836140069.png

2.2 Projeto Java

- Utilizei **Maven** para gerenciar dependências.
- Adicionei o driver do MongoDB no `pom.xml`:

```
<dependency>
    <groupId>org.mongodb</groupId>
    <artifactId>mongodb-driver-sync</artifactId>
    <version>4.11.1</version>
</dependency>
```

3. Implementação do CRUD

3.1 Código Completo

```
import com.mongodb.client.*;
import com.mongodb.client.model.*;
import com.mongodb.client.result.*;
import org.bson.Document;
import org.bson.conversions.Bson;
import java.util.Arrays;
import java.util.Scanner;

public class MongoAtlasCRUD {
    private MongoClient mongoClient;
    private MongoDatabase database;
    private MongoCollection<Document> collection;
    private final Scanner scanner = new Scanner(System.in);

    // Construtor com conexão ao Atlas
    public MongoAtlasCRUD() {
        String connectionString = "mongodb+srv://<user>:
<password>@cluster0.mongodb.net/crudAtlasDB?
retryWrites=true&w=majority";
        this.mongoClient = MongoClients.create(connectionString);
        this.database = mongoClient.getDatabase("crudAtlasDB");
        this.collection = database.getCollection("produtos");
        System.out.println("Conectado ao MongoDB Atlas!");
    }

    // CREATE
    public void criarProduto() {
        System.out.println("\n--- CRIAR PRODUTO ---");
        System.out.print("Nome: ");
        String nome = scanner.nextLine();

        System.out.print("Preço: ");
        double preco = scanner.nextDouble();
```

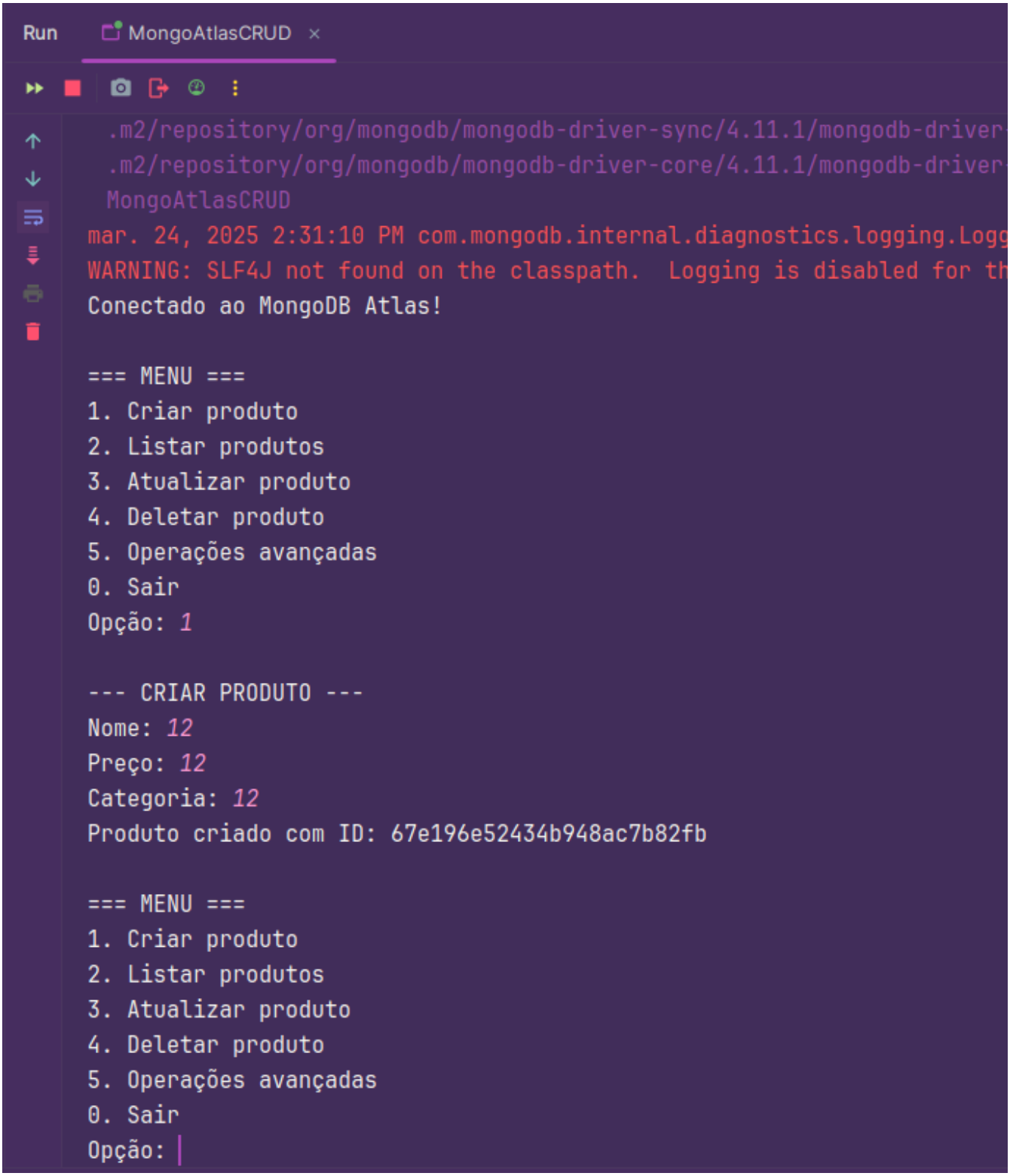
```
        scanner.nextLine();

        System.out.print("Categoria: ");
        String categoria = scanner.nextLine();

        Document produto = new Document("nome", nome)
                .append("preco", preco)
                .append("categoria", categoria);

        collection.insertOne(produto);
        System.out.println("Produto criado com ID: " +
produto.get("_id"));
    }

    // READ
    public void listarProdutos() {
        System.out.println("\n--- LISTAR PRODUTOS ---");
        for (Document produto : collection.find()) {
            System.out.println(produto.toJson());
        }
    }

    // UPDATE
    public void atualizarProduto() {
        System.out.println("\n--- ATUALIZAR PRODUTO ---");
        System.out.print("ID do produto: ");
        String id = scanner.nextLine();

        System.out.print("Novo preço: ");
        double novoPreco = scanner.nextDouble();
        scanner.nextLine();

        Bson filtro = Filters.eq("_id", new
org.bson.types.ObjectId(id));
        Bson atualizacao = Updates.set("preco", novoPreco);

        UpdateResult result = collection.updateOne(filtro,
atualizacao);
        System.out.println(result.getModifiedCount() + " produto(s)
```

```
atualizado(s)");
    }

    // DELETE
    public void deletarProduto() {
        System.out.println("\n--- DELETAR PRODUTO ---");
        System.out.print("ID do produto: ");
        String id = scanner.nextLine();

        Bson filtro = Filters.eq("_id", new
org.bson.types.ObjectId(id));
        DeleteResult result = collection.deleteOne(filtro);

        System.out.println(result.getDeletedCount() + " produto(s)
deletado(s)");
    }

    // OPERAÇÕES AVANÇADAS
    public void operacoesAvancadas() {
        // Agregação: média de preço por categoria
        System.out.println("\nMédia de preço por categoria:");
        collection.aggregate(Arrays.asList(
            Aggregates.group("$categoria",
                Accumulators.avg("mediaPreco", "$preco")
            )
        )).forEach(doc -> System.out.println(doc.toJson()));
    }

    public static void main(String[] args) {
        MongoAtlasCRUD crud = new MongoAtlasCRUD();
        try {
            boolean executando = true;
            while (executando) {
                System.out.println("\n=== MENU ===");
                System.out.println("1. Criar produto");
                System.out.println("2. Listar produtos");
                System.out.println("3. Atualizar produto");
                System.out.println("4. Deletar produto");
                System.out.println("5. Operações avançadas");
```

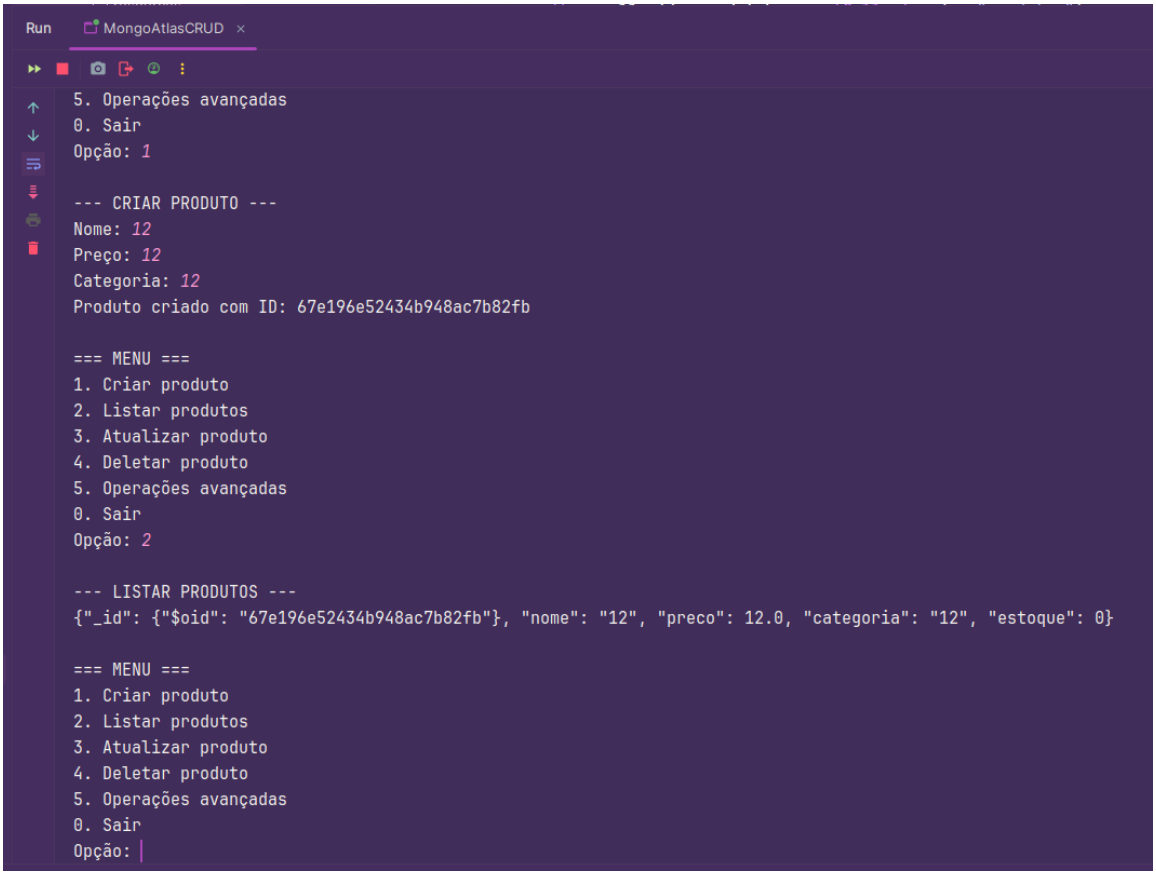
```
                System.out.println("0. Sair");
                System.out.print("Opção: ");

                int opcao =
Integer.parseInt(crud.scanner.nextLine());

                switch (opcao) {
                    case 1 -> crud.criarProduto();
                    case 2 -> crud.listarProdutos();
                    case 3 -> crud.atualizarProduto();
                    case 4 -> crud.deletarProduto();
                    case 5 -> crud.operacoesAvancadas();
                    case 0 -> executando = false;
                    default -> System.out.println("Opção
inválida!");
                }
            }
        } finally {
            crud.mongoClient.close();
        }
    }
}
```

4. Testes Realizados

- Execução do comando de criação de produto no terminal, mostrando o preenchimento dos campos obrigatórios:

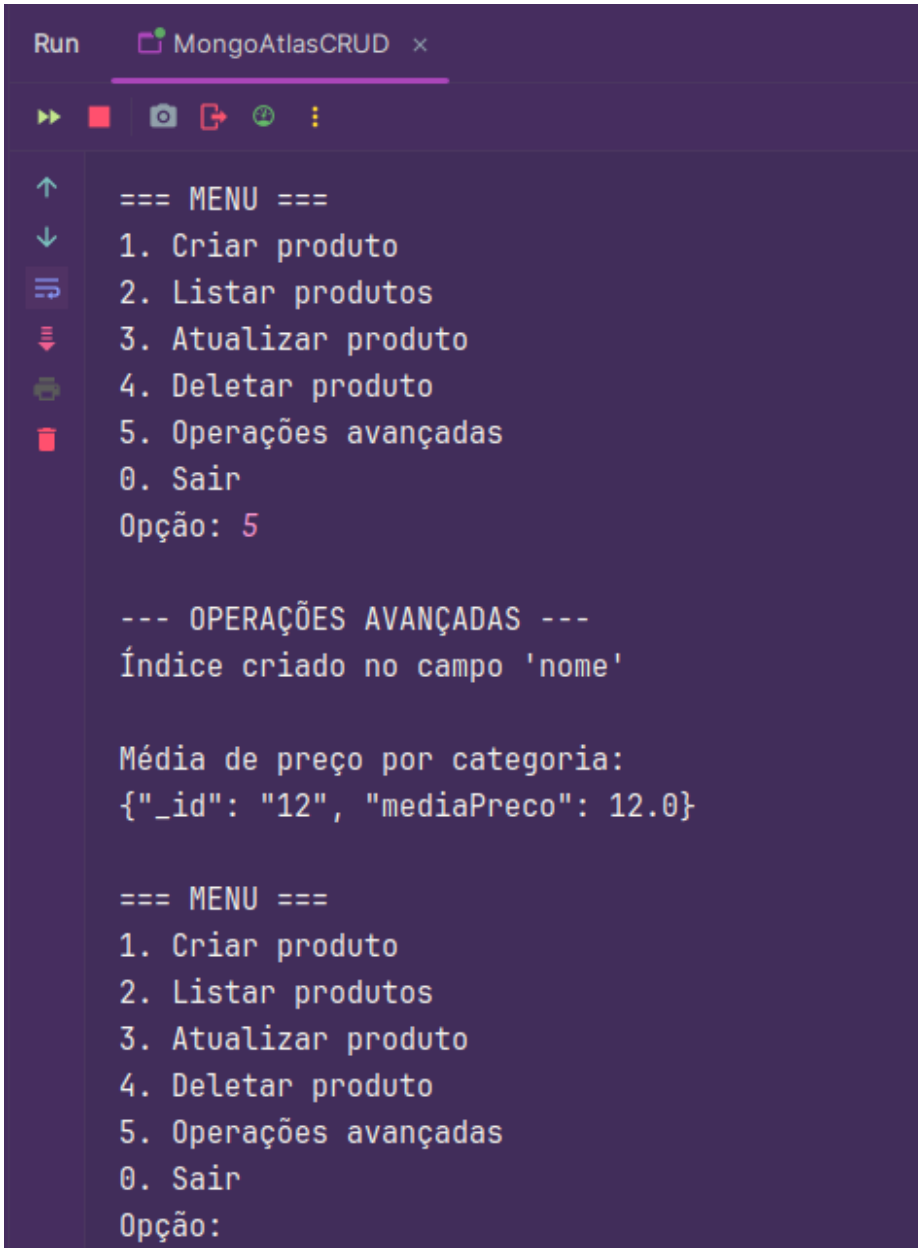


- Resultado da operação de leitura mostrando todos os produtos cadastrados no banco de dados:



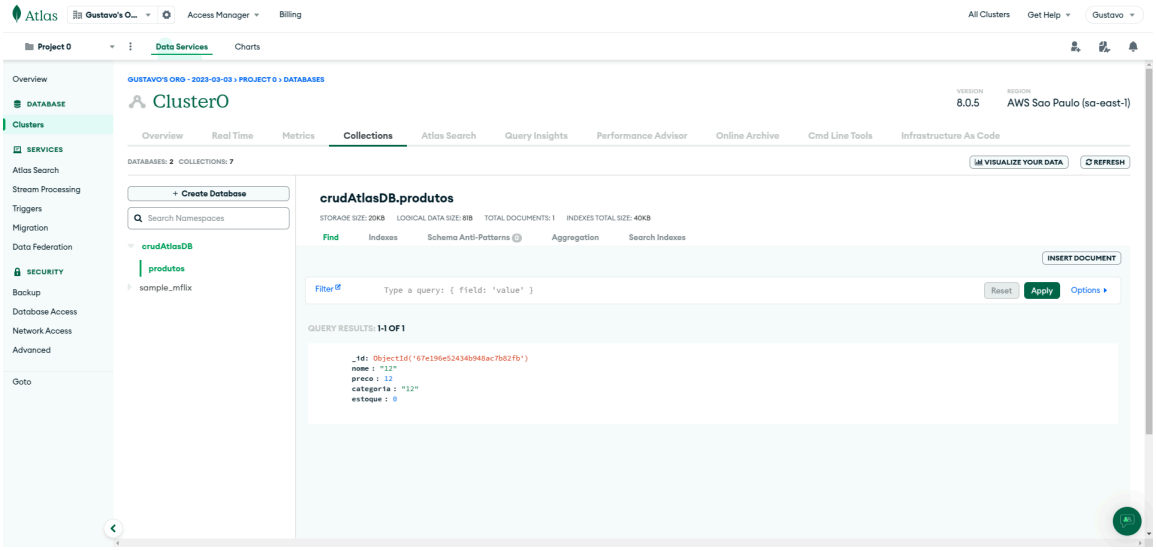
Captura de tela exibindo a listagem completa de produtos em formato JSON após a operação de leitura

- Saída da operação de agregação calculando médias de preço agrupadas por categoria de produtos:



Captura de tela demonstrando a execução da operação avançada de agregação, com cálculo de média de preços por categoria

- Visualização da coleção 'produtos' no MongoDB Atlas comprovando a persistência dos dados inseridos:



Captura de tela do painel do MongoDB Atlas mostrando a coleção 'produtos' com documentos inseridos pelo aplicativo Java

5. Conclusão

Este laboratório demonstrou como conectar uma aplicação Java ao **MongoDB Atlas** e implementar operações **CRUD** com agregações. O código foi testado com sucesso e está pronto para uso em projetos reais.

Referências

Referencias

- MONGODB, Inc. MongoDB Java Driver Documentation. 2023. Disponível em: https://mongodb.github.io/mongo-java-driver/ (https://mongodb.github.io/mongo-java-driver/).
- MONGODB, Inc. MongoDB Atlas Documentation. 2023. Disponível em: https://docs.atlas.mongodb.com/ (https://docs.atlas.mongodb.com/).
- MONGODB University. MongoDB for Java Developers Course. 2023. Disponível em: https://university.mongodb.com/courses/M220J/about (https://university.mongodb.com/courses/M220J/about).
- BAELDUNG. Introduction to MongoDB with Java. 2023. Disponível em: https://www.baeldung.com/java-mongodb (https://www.baeldung.com/java-mongodb).