at_S10_A1_SL06_backend

Roteiro de Atividade Prática

Nome: _____ Turma: _____

Título da atividade: Banco de dados NoSQL

Objetivo:

Implementar e utilizar bancos de dados **NoSQL** para armazenar e recuperar dados de forma flexível e escalável, explorando suas características e diferenças em relação aos bancos de dados relacionais.

Conceito técnico: banco de dados NoSQL

Os **bancos de dados NoSQL** (Not Only SQL) são sistemas de gerenciamento de banco de dados que fogem da estrutura tradicional de tabelas e relações presentes nos bancos de dados relacionais. Eles são otimizados para cenários em que a escalabilidade, a flexibilidade e o alto volume de dados são essenciais.

Alguns dos tipos mais comuns de bancos de dados NoSQL incluem:

- **Document Store (MongoDB):** armazena dados em documentos, geralmente no formato JSON, permitindo uma estrutura flexível de campos;
- **Key-Value Store (Redis):** armazena dados como pares chave-valor, sendo ideal para caches e dados simples;
- **Column Family Store (Cassandra):** usa colunas para organizar dados, facilitando a consulta de grandes volumes de informações;
- **Graph Database (Neo4j):** armazena dados em nós e relacionamentos, sendo eficiente para consultas de grafos.

NoSQL é frequentemente usado em sistemas que precisam de alta disponibilidade e escalabilidade horizontal, como grandes sistemas distribuídos e aplicações que lidam com big data.

Objetivo da atividade:

O participante irá implementar um banco de dados NoSQL utilizando **MongoDB** para gerenciar produtos de um sistema de e-commerce. A API deve permitir que o usuário adicione, visualize e remova produtos utilizando operações CRUD (create, read, update, delete).

Enunciado:

Você foi encarregado de implementar um banco de dados NoSQL utilizando **MongoDB** para armazenar e gerenciar os produtos de um sistema de e-commerce. Sua API deve permitir as seguintes operações:

- adicionar um novo produto ao banco de dados;
- visualizar todos os produtos ou um produto específico;
- atualizar e remover produtos.

Implemente a solução utilizando **Python** e a biblioteca **pymongo** para interagir com o MongoDB.

Código de programação:

Instalação do MongoDB e pymongo:

1. **Instalar o MongoDB:**
   Siga as instruções para instalar o MongoDB localmente ou utilize um serviço de banco de dados MongoDB na nuvem (como MongoDB Atlas).
2. **Instalar a biblioteca pymongo:**

pip install pymongo

Exemplo em Python com pymongo:

```
from pymongo import MongoClient
from flask import Flask, jsonify, request

app = Flask(__name__)

# Conexão com o banco de dados MongoDB
client = MongoClient('mongodb://localhost:27017/')
db = client['ecommerce']
produtos_collection = db['produtos']

# Rota para adicionar um novo produto (CREATE)
@app.route('/produtos', methods=['POST'])
def adicionar_produto():
    novo_produto = request.get_json()
    produto_id = produtos_collection.insert_one(novo_produto).inserted_id
    return jsonify({"mensagem": "Produto adicionado com sucesso!", "id":
str(produto_id)}), 201

# Rota para listar todos os produtos (READ)
@app.route('/produtos', methods=['GET'])
def listar_produtos():
    produtos = list(produtos_collection.find())
    for produto in produtos:
        produto['_id'] = str(produto['_id'])  # Converter ObjectId para string
    return jsonify(produtos)

# Rota para buscar um produto por ID (READ)
@app.route('/produtos/<id>', methods=['GET'])
def obter_produto(id):
    produto = produtos_collection.find_one({"_id": ObjectId(id)})
    if produto:
        produto['_id'] = str(produto['_id'])
        return jsonify(produto)
    else:
        return jsonify({"erro": "Produto não encontrado"}), 404
```

```
# Rota para atualizar um produto (UPDATE)
@app.route('/produtos/<id>', methods=['PUT'])
def atualizar_produto(id):
    dados_atualizados = request.get_json()
    resultado = produtos_collection.update_one({"_id": ObjectId(id)}, {"$set":
dados_atualizados})
    if resultado.matched_count > 0:
        return jsonify({"mensagem": "Produto atualizado com sucesso!"})
    else:
        return jsonify({"erro": "Produto não encontrado"}), 404

# Rota para remover um produto (DELETE)
@app.route('/produtos/<id>', methods=['DELETE'])
def remover_produto(id):
    resultado = produtos_collection.delete_one({"_id": ObjectId(id)})
    if resultado.deleted_count > 0:
        return jsonify({"mensagem": "Produto removido com sucesso!"})
    else:
        return jsonify({"erro": "Produto não encontrado"}), 404

if __name__ == '__main__':
    app.run(debug=True)
```

Explicação técnica:

- **Conexão com o MongoDB:** a aplicação se conecta ao MongoDB utilizando a biblioteca **pymongo**. O banco de dados **ecommerce** é usado para armazenar os produtos.
- **Operações CRUD:** a aplicação suporta operações CRUD:
    - **POST /produtos:** insere um novo produto no banco;
    - **GET /produtos:** lista todos os produtos ou recupera um produto específico por ID;
    - **PUT /produtos/{id}:** atualiza um produto existente com os dados enviados;
    - **DELETE /produtos/{id}:** remove um produto por ID.

- **Documentos em MongoDB:** cada produto é armazenado como um documento no formato JSON, permitindo flexibilidade na estrutura dos dados.

Perguntas para conclusão da atividade:

- Explique por que bancos de dados NoSQL como MongoDB são mais adequados para sistemas com grandes volumes de dados e necessidade de alta escalabilidade.

- Descreva a diferença entre o armazenamento de dados em MongoDB e um banco de dados relacional como MySQL.
-
- Como o uso de ObjectId em MongoDB ajuda a identificar documentos de forma única e eficiente?