Trabalho Final Banco de Dados II

📝 README – Trabalho Final de Banco de Dados II

Integração: PostgreSQL + MongoDB + Neo4j + Redis + Node.js

📌 1. Visão Geral do Projeto

Este projeto consiste na implementação de uma **API Node.js** integrada com **quatro tipos de bancos de dados**:

- 🟦 **PostgreSQL** – Banco relacional (clientes, produtos, compras)
- 🟩 **MongoDB** – Banco de documentos (interesses dos clientes)
- 🟪 **Neo4j** – Banco de grafos (relacionamentos de amizade entre clientes)
- 🟥 **Redis** – Camada de cache

O objetivo do trabalho é demonstrar, na prática, o uso e a integração de múltiplos modelos de bancos de dados dentro de um único sistema.

A API permite consultar clientes, compras, interesses, relacionamentos e recomendações, utilizando o melhor tipo de banco para cada tipo de dado.

📂 2. Estrutura do Projeto

```
src/
  app.js
  config/
    db.js
    mongoClient.js
    neo4jClient.js
    redisClient.js
```

```
  database/
    graphService.js
    documentService.js
  services/
    relationalService.js
  routes/
    relationalRoutes.js
    mongoRoutes.js
    graphRoutes.js
    redisRoutes.js
.env
package.json
```

🚀 3. Tecnologias Utilizadas

Tecnologia	Função
Node.js + Express	Backend/API
PostgreSQL	Banco relacional
MongoDB	Banco NoSQL de documentos
Neo4j	Banco de grafos
Redis	Cache em memória
pg	Cliente PostgreSQL
mongodb	Cliente Mongo
neo4j-driver	Driver oficial Neo4j
redis	Cliente Redis

⚙️ 4. Como Rodar o Projeto

1️⃣ Instalar dependências

```
npm install
```

2️⃣ Configurar o .env

Crie um arquivo `.env` baseado no `.env.example`:

```
PGUSER=maria
PGHOST=localhost
PGDATABASE=trabalho_final_banco_2
PGPASSWORD=753495
PGPORT=5432

MONGO_URI=mongodb://localhost:27017
MONGO_DB_NAME=trabalho_final_bd
MONGO_COLLECTION_INTERESSES=interesses_clientes

NEO4J_URI=bolt://localhost:7687
NEO4J_USER=neo4j
NEO4J_PASSWORD=mariahoppe

REDIS_HOST=127.0.0.1
REDIS_PORT=6379

PORT=3001
```

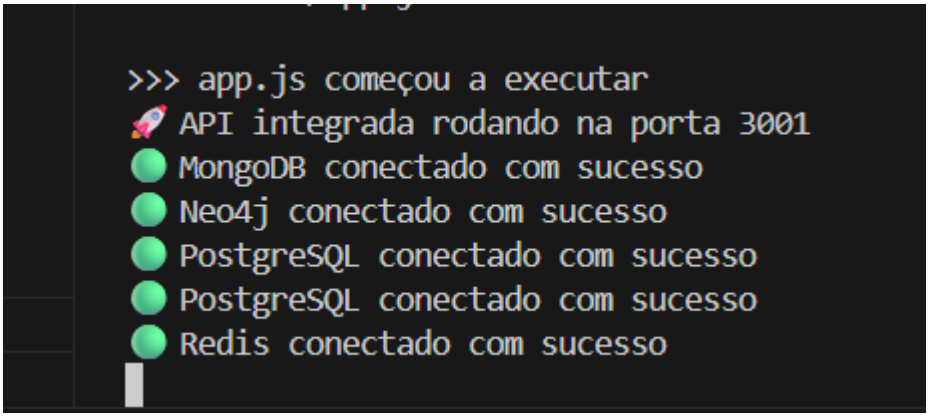
3️⃣ Iniciar a API

```
npm run dev
```

4️⃣ Saída esperada:

```
🟢 MongoDB conectado com sucesso
🟢 Neo4j conectado com sucesso
🟢 PostgreSQL conectado com sucesso
🟢 Redis conectado com sucesso
🚀 API integrada rodando na porta 3001
```

📌



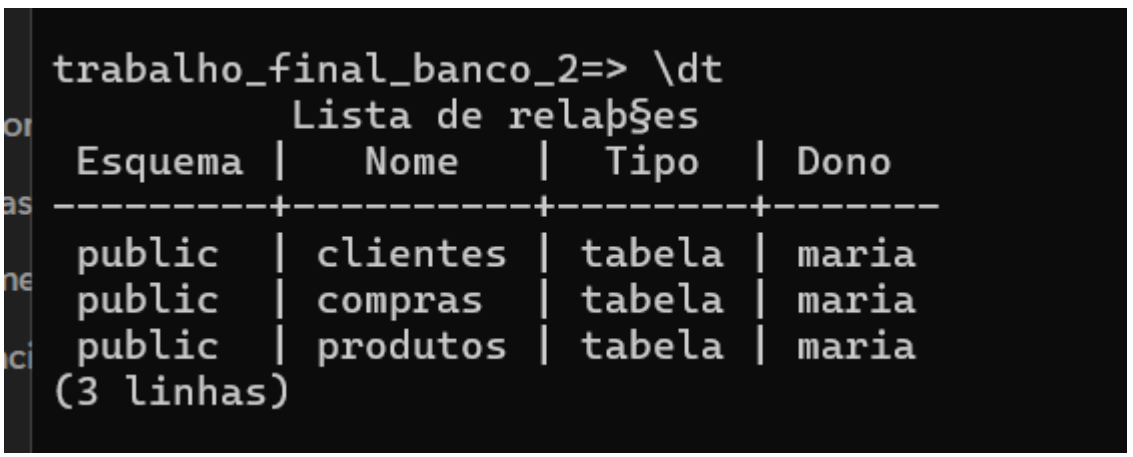
🗄️ 5. Modelagem dos Bancos

5.1 – PostgreSQL (Relacional)

Tabelas criadas

- clientes
- produtos
- compras

📌



Script SQL

Inclua aqui o script completo de criação das tabelas e inserts.

```
-- ================================================
-- CRIAÇÃO DAS TABELAS
```

```
-- ================================================

DROP TABLE IF EXISTS compras CASCADE;
DROP TABLE IF EXISTS produtos CASCADE;
DROP TABLE IF EXISTS clientes CASCADE;

-- CLIENTES
CREATE TABLE clientes (
    id SERIAL PRIMARY KEY,
    cpf VARCHAR(14) NOT NULL UNIQUE,
    nome VARCHAR(100) NOT NULL,
    cidade VARCHAR(100),
    uf CHAR(2),
    email VARCHAR(100)
);

-- PRODUTOS
CREATE TABLE produtos (
    id SERIAL PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    preco NUMERIC(10,2) NOT NULL
);

-- COMPRAS
CREATE TABLE compras (
    id SERIAL PRIMARY KEY,
    id_cliente INTEGER NOT NULL,
    id_produto INTEGER NOT NULL,
    data DATE NOT NULL,
    valor NUMERIC(10,2) NOT NULL,
    FOREIGN KEY (id_cliente) REFERENCES clientes(id),
    FOREIGN KEY (id_produto) REFERENCES produtos(id)
);

-- ================================================
-- INSERT DOS 10 CLIENTES
-- ================================================
```

```
INSERT INTO clientes (id, cpf, nome, cidade, uf, email) VALUES
(1, '00011122233', 'Maria Silva', 'Chapecó', 'SC', 'maria@gmail.com'),
(2, '11122233344', 'João Souza', 'Florianópolis', 'SC', 'joao@gmail.com'),
(3, '22233344455', 'Carla Lima', 'Chapecó', 'SC', 'carla@gmail.com'),
(4, '33344455566', 'Daniel Costa', 'Porto Alegre', 'RS', 'daniel@gmail.co
m'),
(5, '44455566677', 'Eduarda Pereira', 'Xaxim', 'SC', 'eduarda@gmail.com'),
(6, '55566677788', 'Felipe Oliveira', 'Chapecó', 'SC', 'felipe@gmail.com'),
(7, '66677788899', 'Gabriela Santos', 'Curitiba', 'PR', 'gabi@gmail.com'),
(8, '77788899900', 'Henrique Almeida', 'Maravilha', 'SC', 'henrique@gmail.
com'),
(9, '88899900011', 'Isabela Rocha', 'São Paulo', 'SP', 'isabela@gmail.com'),
(10,'99900011122', 'João Martins', 'Chapecó', 'SC', 'joaom@gmail.com');

-- ================================================
-- INSERT DOS 19 PRODUTOS (IDs 1–19)
-- ================================================

INSERT INTO produtos (id, nome, preco) VALUES
(1, 'Notebook Lenovo', 3500.00),
(2, 'Mouse Gamer', 120.00),
(3, 'Teclado Mecânico', 250.00),
(4, 'Monitor 24"', 899.90),
(5, 'Cadeira Gamer', 1200.00),
(6, 'Headset', 199.90),
(7, 'HD 1TB', 320.00),
(8, 'SSD 480GB', 280.00),
(9, 'Placa de Vídeo GTX 1650', 1600.00),
(10,'Processador Ryzen 5', 950.00),
(11,'Memória RAM 8GB', 180.00),
(12,'Smartphone Samsung', 2500.00),
(13,'Tablet Lenovo', 1100.00),
(14,'Webcam FullHD', 150.00),
(15,'Microfone Condensador', 300.00),
(16,'Carregador Turbo', 90.00),
(17,'Cabo HDMI', 40.00),
(18,'Caixa de Som JBL', 399.00),
(19,'Estabilizador', 250.00);
```

```
-- ================================================
-- INSERT DAS 20 COMPRAS
-- ================================================

INSERT INTO compras (id_cliente, id_produto, data, valor) VALUES
(1, 1, '2025-01-11', 3500.00),
(1, 5, '2025-01-12', 1200.00),
(2, 2, '2025-01-10', 120.00),
(2, 10, '2025-01-09', 950.00),
(3, 12, '2025-02-01', 2500.00),
(3, 17, '2025-02-02', 40.00),
(4, 3, '2025-02-10', 250.00),
(4, 11, '2025-02-11', 180.00),
(5, 8, '2025-03-01', 280.00),
(5, 18, '2025-03-02', 399.00),
(6, 4, '2025-03-10', 899.90),
(6, 7, '2025-03-11', 320.00),
(7, 13, '2025-03-14', 1100.00),
(7, 6, '2025-03-15', 199.90),
(8, 9, '2025-03-20', 1600.00),
(8, 16, '2025-03-21', 90.00),
(9, 14, '2025-03-25', 150.00),
(9, 19, '2025-03-26', 250.00),
(10, 15, '2025-04-01', 300.00),
(10, 1, '2025-04-02', 3500.00);
```

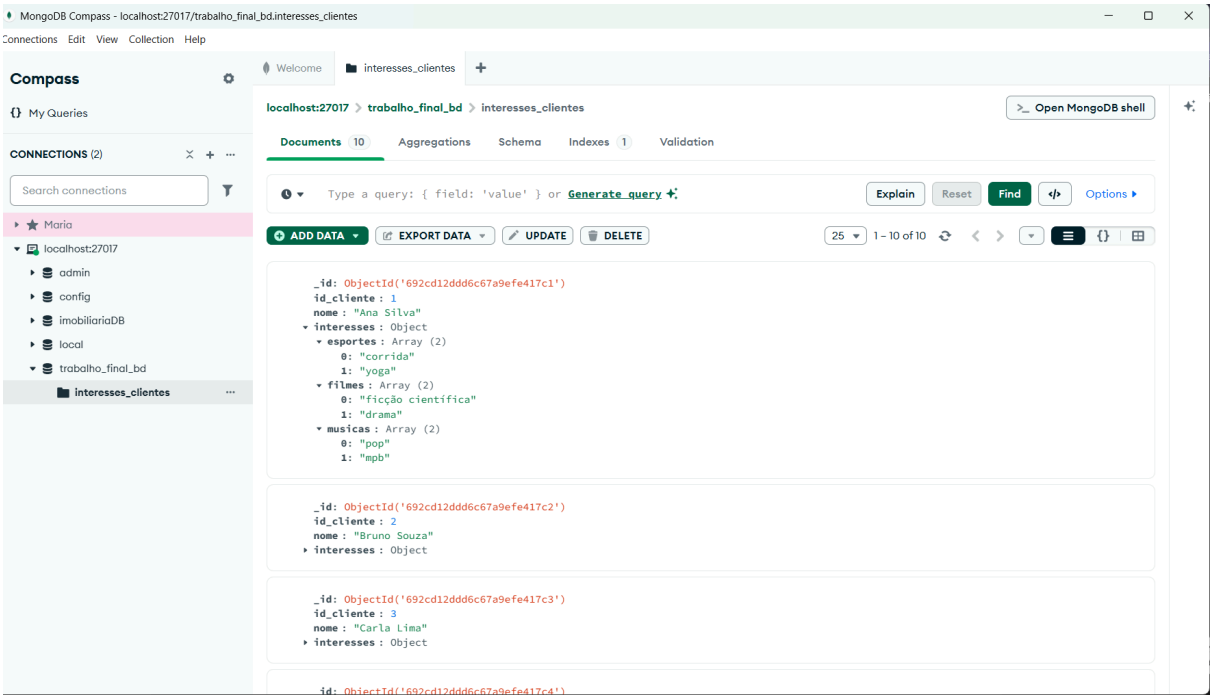
5.2 – MongoDB (Documentos)

Armazena interesses de cada cliente:

```
{
  "id_cliente": 1,
  "nome": "Ana Silva",
  "interesses": {
    "esportes": ["corrida", "yoga"],
    "filmes": ["ficção científica", "drama"],
    "musicas": ["pop", "mpb"]
```

```
  }
}
```

📌



5.3 – Neo4j (Grafos)

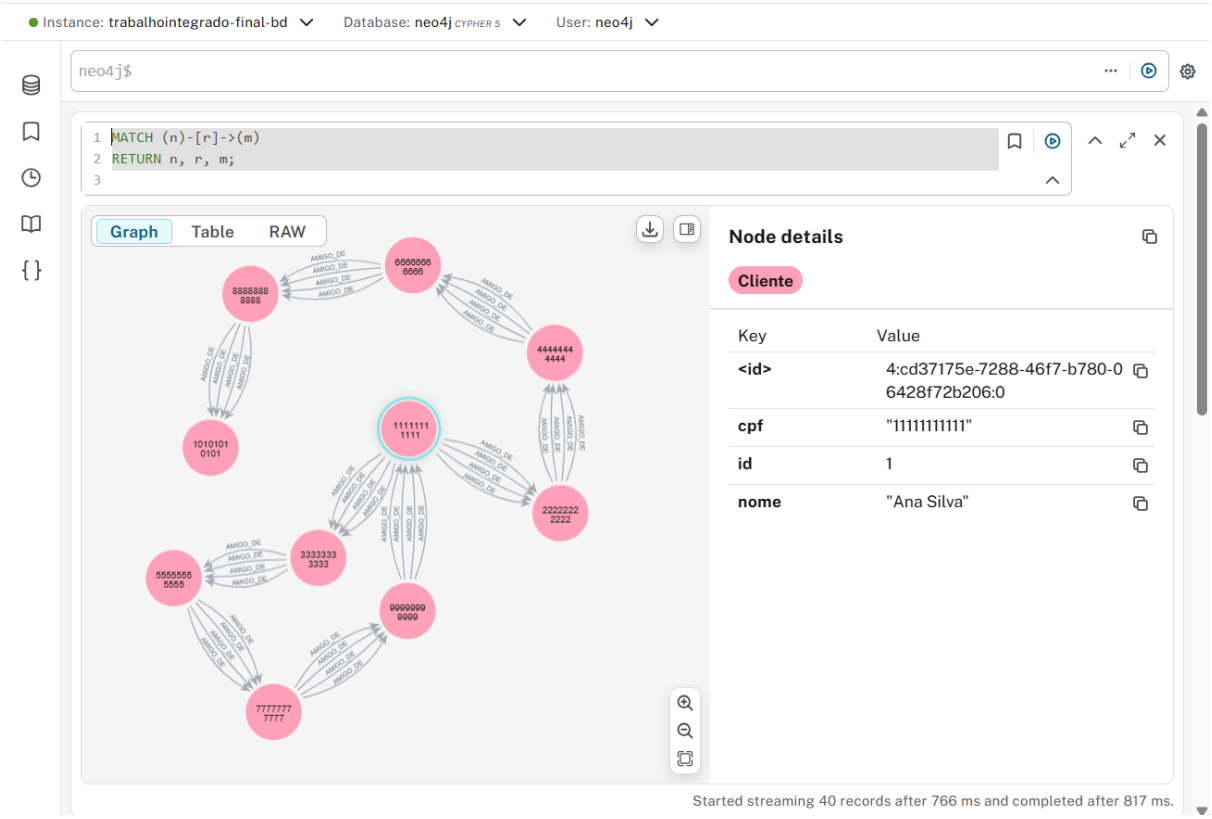
Cada cliente é um nó: `(Cliente)`

E os relacionamentos são:

```
(c1)-[:AMIGO_DE]→(c2)

MATCH (n)-[r]→(m)
RETURN n, r, m;
```

📌



5.4 – Redis (Cache)

Usado para guardar:

- lista de clientes
- lista de interesses
- sincronização de dados

🔌 6. Rotas da API

⭐ 6.1 – PostgreSQL

Método	Rota	Descrição
GET	`/clientes`	Lista todos os clientes
GET	`/compras/:id`	Lista compras de um cliente
GET	`/clientes-compras`	Clientes + compras (JOIN)

⭐ 6.2 – MongoDB

Método	Rota	Descrição
GET	`/mongo/interesses`	Lista interesses
GET	`/mongo/interesses/:id`	Busca interesses de um cliente

⭐ 6.3 – Neo4j

Método	Rota	Descrição
GET	`/neo/clientes-amigos`	Lista todos os clientes + amigos
GET	`/neo/clientes-amigos/:id`	Lista amigos de um cliente
GET	`/neo/recomendacoes/:id`	Recomenda novos amigos

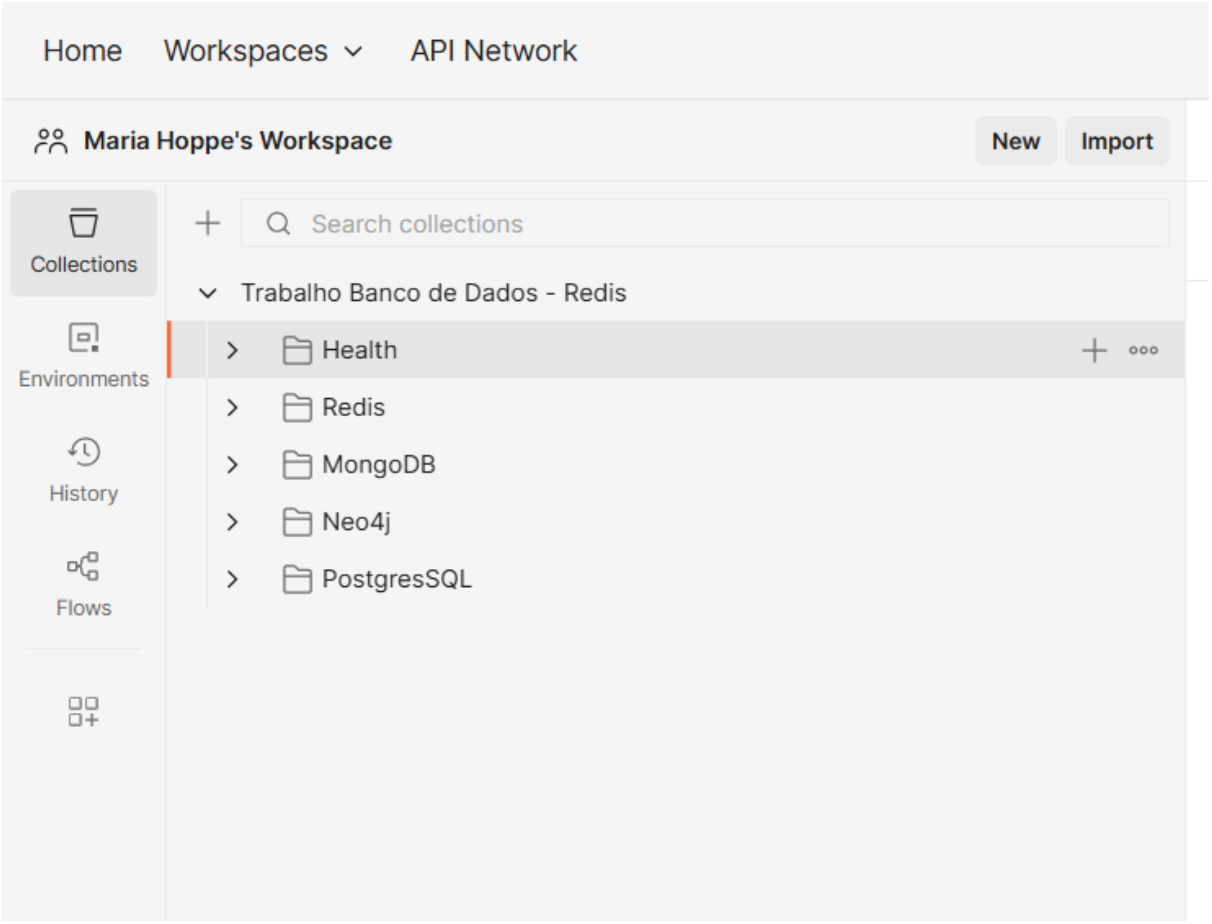
⭐ 6.4 – Redis

Método	Rota	Descrição
GET	`/redis/clientes`	Lista clientes pelo cache
GET	`/redis/clientes-compras`	Mostrar compras do cliente
GET	`/redis/cliente-amigos`	Mostra os dados do cliente e amigos
GET	`/redis/amigos-recomendacoes`	Mostra as recomendações dos amigos
GET	`/sync`	Sincroniza Redis ⇄ Mongo

📫 7. Testes no Postman

Crie uma coleção contendo todas as rotas acima.

https://maria-hoppe-8931454.postman.co/workspace/87ceb8db-c761-487b-8179-87240646dc87/folder/50419383-0dc27f8c-1ef8-4124-b2ee-b78f7c070e51



🏁 8. Conclusão

Este trabalho demonstra como integrar múltiplos bancos de dados — relacionais, documentos, grafos e memória — em um único sistema Node.js, utilizando:

- consultas SQL
- consultas NoSQL
- consultas em Grafos
- caching em Redis
- boas práticas de arquitetura

Foi implementado um backend completo, modular e funcional, que permite operações reais envolvendo dados diferentes em bancos diferentes, provando a versatilidade da API.

.env.example

```
# ------------------------------
# ⚙️ CONFIGURAÇÕES POSTGRESQL
# ------------------------------
PGUSER=
PGHOST=localhost
PGDATABASE=
PGPASSWORD=
PGPORT=5432

PGPOOL_MAX=10
PGPOOL_IDLE=30000

# Porta da API
PORT=3001

# ------------------------------
# 🍃 CONFIGURAÇÕES MONGODB
# ------------------------------
MONGO_URI=mongodb://localhost:27017
MONGO_DB_NAME=
MONGO_COLLECTION_INTERESSES=interesses_clientes

# ------------------------------
# 🟣 CONFIGURAÇÕES NEO4J
# ------------------------------
NEO4J_URI=bolt://localhost:7687
NEO4J_USER=
NEO4J_PASSWORD=

# ------------------------------
# 🔴 CONFIGURAÇÕES REDIS
# ------------------------------
REDIS_HOST=127.0.0.1
REDIS_PORT=6379
```

✔️ O que o aluno deve preencher depois de baixar o projeto

- Usuário do PostgreSQL (PGUSER)
- Banco do PostgreSQL (PGDATABASE)
- Senha do PostgreSQL (PGPASSWORD)
- Nome do banco no Mongo (MONGO_DB_NAME)
- Usuário e senha do Neo4j (caso use senha)
- Se usar Redis externo, ajustar host/porta

✔️ 9. Créditos

Projeto desenvolvido por:

Guilherme Moreira Casagrande, Maria Fernanda Henker Hoppe e Mário Laux Neto

Disciplina: Banco de Dados II – UNOCHAPECÓ

Professora: **Monica Tissiani de Toni Pereira**

Link repositório Github : https://github.com/mariahoppe/trabalointegrado-final-bd

---