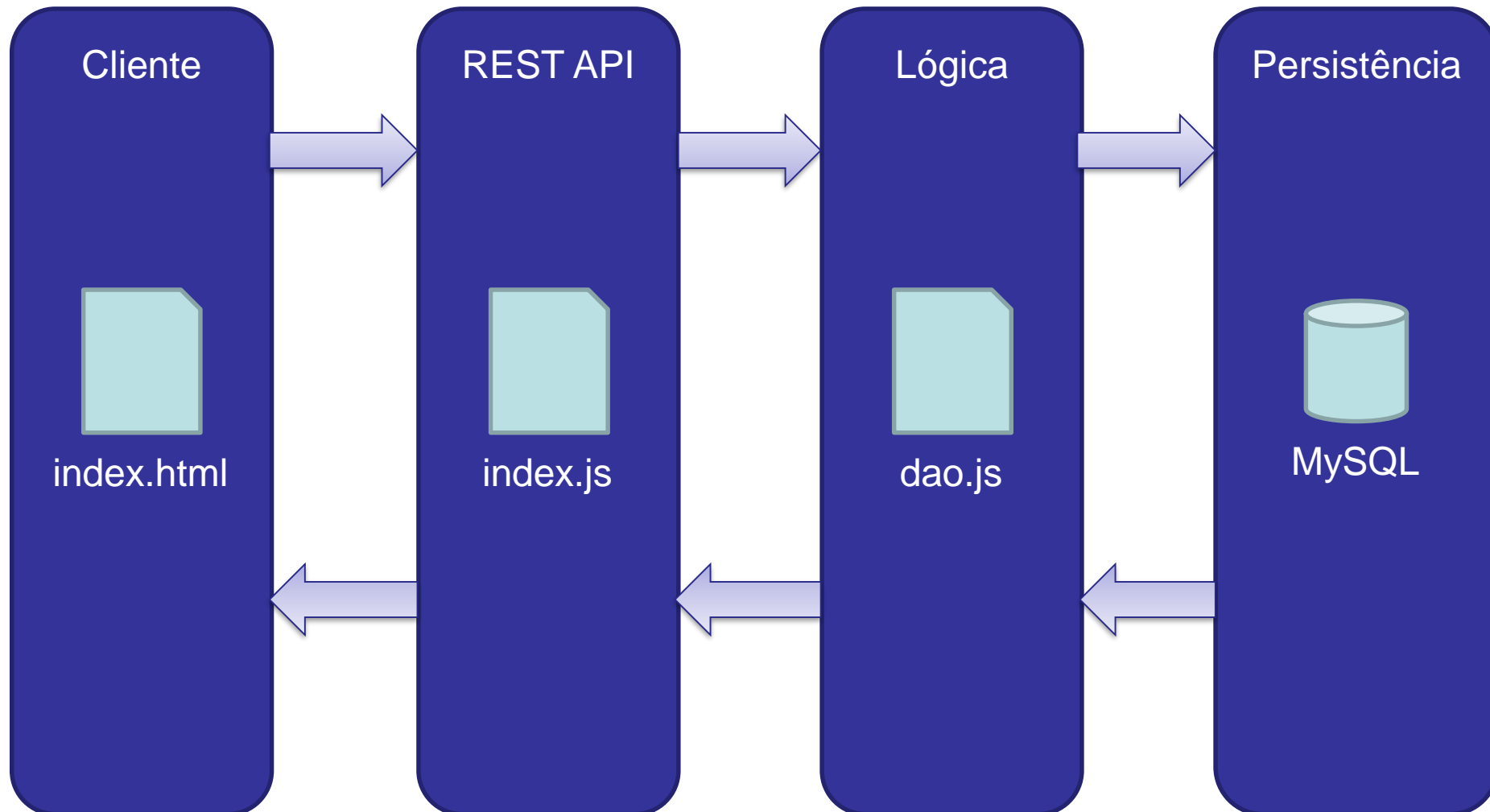


Trocando Persistência para MongoDB

EC021 - Tópicos Avançados II
Sistemas Distribuídos

Introdução

- Na práticas anteriores separamos a camada de lógica do sistema da camada de acesso ao servidor (Rest API).



O MongoDB

- O que é?
 - Banco de dados (BD) não relacional orientado a documentos.
- Não relacional?
 - Não existe a necessidade de criar uma estrutura de tabelas e dados antes de começar a inserir informações.
- Pra que serve?
 - Mesmo tendo um conceito diferente, é um banco de dados: serve para armazenar informações.
 - Utilizado quando o modelo estrutural não é adequado.
 - Cada banco de dados é mais adequado pra cada situação.

O MongoDB

- Alguns Recursos
 - Alta disponibilidade.
 - Escalabilidade horizontal.
 - Armazenamento de arquivos (binários).
 - Agregation framework (Trabalhar com estatísticas, cruzamento de informações).
 - Dar suporte a diversas linguagens de programação.
 - C#, Java, Php, Python, NodeJs.
 - Joins entre coleções.

O MongoDB

- Em relação à um BD relacional, podemos fazer a seguinte comparação:

BD Relacional	BD Orientado a Documentos
Tabelas	Coleções (Collections)
Linhas	Documentos
Colunas	Campos

O MongoDB

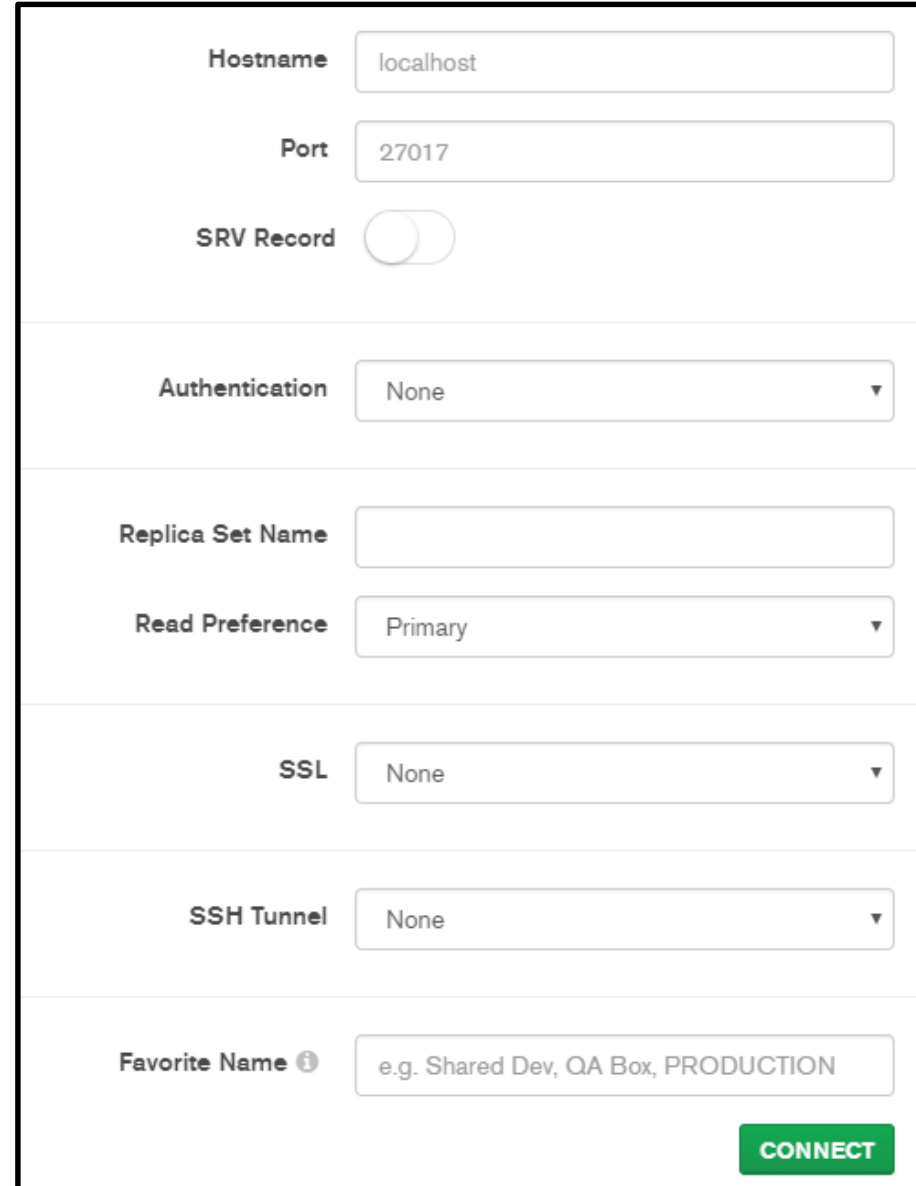
- Conta com as seguintes funcionalidades:
 - Query Language ou Mongo Shell
 - MongoDB: Serviço do MongoDB
 - Replicação
 - Alta disponibilidade
 - Primário – Secundário – Secundário
 - Sempre que o primário cair, um secundário assume
 - Todas as requisições são feitas no primário

O MongoDB



Criando o BD

- Abra o MongoDB Compass e configure uma “New Connection”:

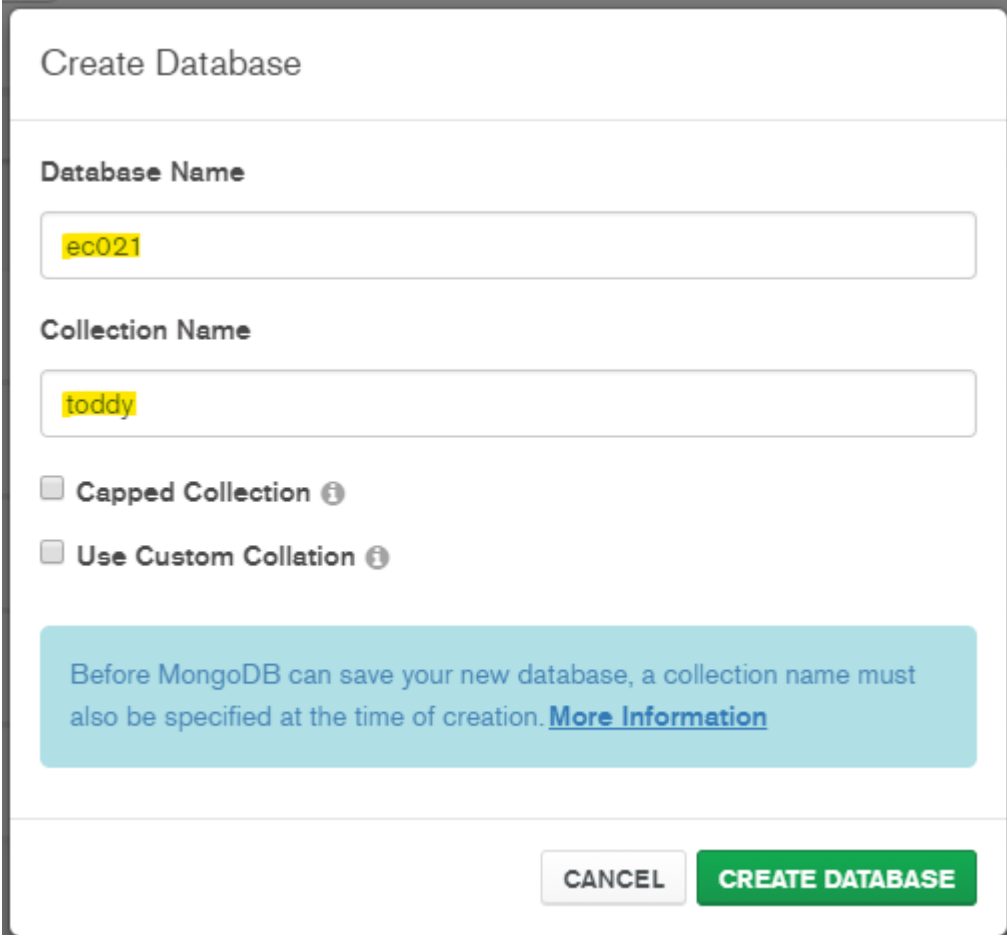


The image shows the 'New Connection' configuration form in MongoDB Compass. The form is organized into several sections with labels on the left and input fields on the right. The 'Hostname' field contains 'localhost'. The 'Port' field contains '27017'. The 'SRV Record' is a toggle switch that is currently turned off. The 'Authentication' dropdown menu is set to 'None'. The 'Replica Set Name' field is empty. The 'Read Preference' dropdown menu is set to 'Primary'. The 'SSL' dropdown menu is set to 'None'. The 'SSH Tunnel' dropdown menu is set to 'None'. The 'Favorite Name' field contains the text 'e.g. Shared Dev, QA Box, PRODUCTION'. At the bottom right, there is a green 'CONNECT' button.

Hostname	localhost
Port	27017
SRV Record	<input type="checkbox"/>
Authentication	None
Replica Set Name	
Read Preference	Primary
SSL	None
SSH Tunnel	None
Favorite Name ⓘ	e.g. Shared Dev, QA Box, PRODUCTION
CONNECT	

Criando o BD

- Após conectado selecione a opção “Create Database”.
- Na janela popup que vai se abrir entre com o nome do BD e da primeira coleção a ser criada:



The screenshot shows the 'Create Database' dialog box in MongoDB. It has a title bar 'Create Database'. Below it, there are two text input fields: 'Database Name' with the value 'ec021' and 'Collection Name' with the value 'toddy'. Below these fields are two checkboxes: 'Capped Collection' and 'Use Custom Collation', both of which are unchecked. At the bottom of the dialog, there is a light blue informational box that reads: 'Before MongoDB can save your new database, a collection name must also be specified at the time of creation. [More Information](#)'. At the very bottom, there are two buttons: 'CANCEL' and 'CREATE DATABASE'.

Create Database

Database Name

ec021

Collection Name

toddy

☐ Capped Collection ⓘ

☐ Use Custom Collation ⓘ

Before MongoDB can save your new database, a collection name must also be specified at the time of creation. [More Information](#)

CANCEL CREATE DATABASE

Instalando Biblioteca do MongoDB

- Abra um prompt na pasta do seu backend e instale a biblioteca do MongoDB:

```
npm install mongodb
```

- Como a persistência será alterada, podemos remover a biblioteca do mysql:

```
npm remove mysql
```

Alterando o dao.js

- Agora faremos as alterações necessárias no nosso módulo de persistência para se conectar no MongoDB.
- **Importante:** a ideia de modularizar o sistema e se trabalhar com projeto baseado em componentes é exatamente tornar possível alterar um modulo sem que haja impacto nos demais módulos (por exemplo, no index.js).
- Importe os módulos abaixo do MongoDB:

```
/**
 * Classes da biblioteca do MongoDB
 */
const MongoClient = require('mongodb').MongoClient;
//ObjectId => Usado para referências ao ID das entidades da coleção
const ObjectId = require('mongodb').ObjectId;
```

Alterando o dao.js

- Configure a conexão com seu BD local do MongoDB:

```
const uri = 'mongodb://127.0.0.1:27017/'; //Endereço do nosso server
const params = {
  //Parâmetro necessário para versões mais novas do MongoDB
  useNewUrlParser: true
};
const dbName = 'ec021'; //Nome do nosso BD
const collName = 'toddy'; //Nome da coleção do nosso DB
```

Conectando no MongoDB

- Dentro de cada função de persistência do seu backend a conexão com o BD deve seguir o formato abaixo:

```
//Abrindo a conexão com o MongoDB
MongoClient.connect(uri, params, function (err, client) {
  if (!err) {
    //Operações no BD
  } else {
    //Devolvemos um erro caso haja erro na conexão com o MongoDB
  }
});
```

Operações no BD

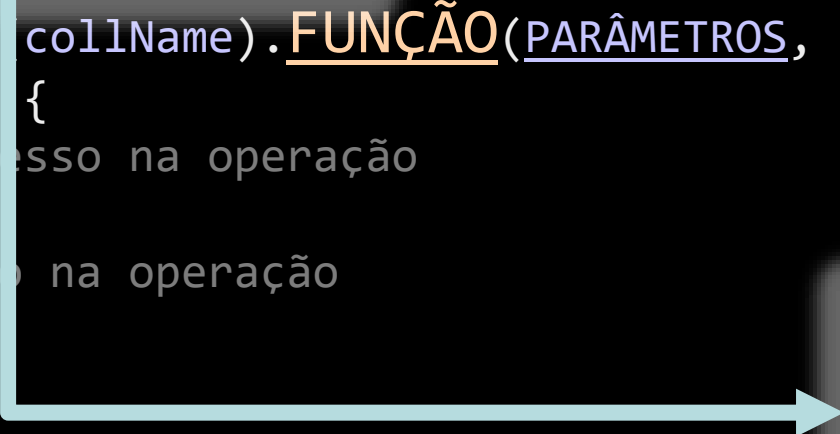
- O formato dos métodos que farão interação com o BD ficará assim:

```
//Abrindo a conexão com o MongoDB
MongoClient.connect(uri, params, function (err, client) {
  if (!err) {
    var db = client.db(dbName); //Selecionando o banco para conectar
    db.collection(collName).FUNÇÃO(PARÂMETROS, CALLBACK () {
      if (!err) {
        //Sucesso na operação
      } else {
        //Erro na operação
      }
    });
  } else {
    //Erro na conexão com o MongoDB
  }
});
```

Operações no BD

- O formato dos métodos que farão interação com o BD ficará assim:

```
//Abrindo a conexão com o MongoDB
MongoClient.connect(uri, params, function (err, client) {
  if (!err) {
    var db = client.db(dbName); //Selecionando o banco para conectar
    db.collection(collName).FUNÇÃO(PARÂMETROS, CALLBACK () {
      if (!err) {
        //Sucesso na operação
      } else {
        //Erro na operação
      }
    });
  } else {
    //Erro na conexão com o MongoDB
  }
});
```

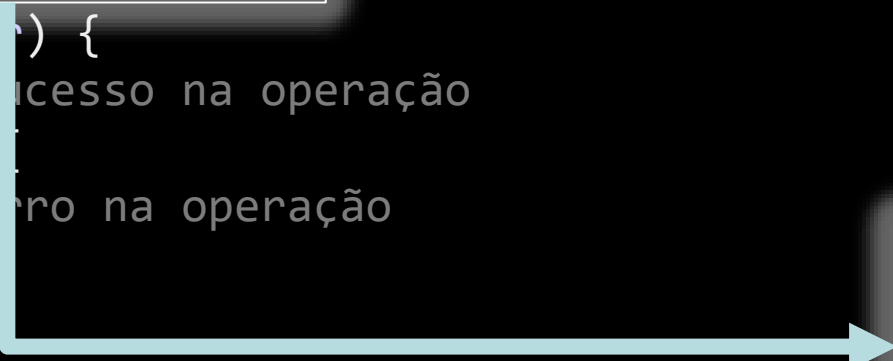


Usando o 'client' para selecionar o BD que vamos usar.

Operações no BD

- O formato dos métodos que farão interação com o BD ficará assim:

```
//Abrindo a conexão com o MongoDB
MongoClient.connect(uri, params, function (err, client) {
  if (!err) {
    var db = client.db(dbName); //Selecionando o banco para conectar
    db.collection(collName).FUNÇÃO(PARÂMETROS, CALLBACK () {
      if (!err) {
        //Sucesso na operação
      } else {
        //Erro na operação
      }
    });
  } else {
    //Erro na conexão com o MongoDB
  }
});
```

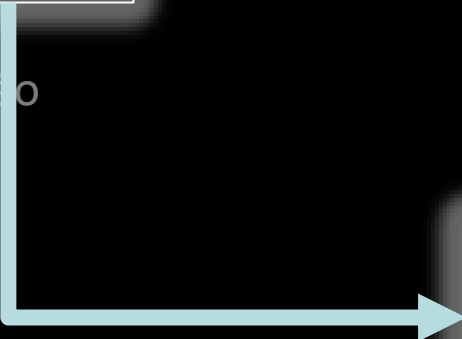


Usando o 'db' para selecionar a coleção que vamos usar.

Operações no BD

- O formato dos métodos que farão interação com o BD ficará assim:

```
//Abrindo a conexão com o MongoDB
MongoClient.connect(uri, params, function (err, client) {
  if (!err) {
    var db = client.db(dbName); //Selecionando o banco para conectar
    db.collection(collName).FUNÇÃO(PARÂMETROS, CALLBACK () {
      if (!err) {
        //Sucesso na operação
      } else {
        //Erro na operação
      }
    });
  } else {
    //Erro na conexão com o MongoDB
  }
});
```



Aqui definimos qual operação será executada no BD.

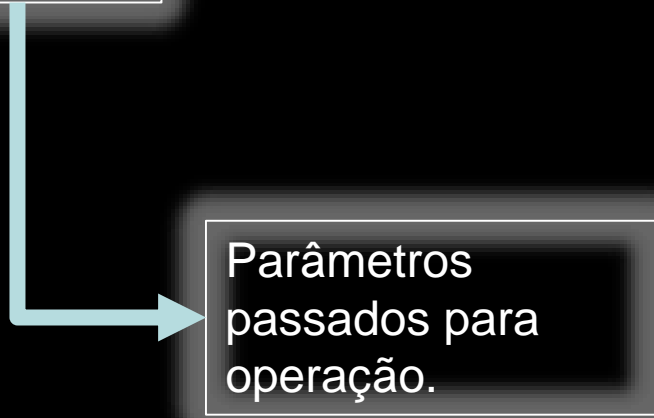
Operações de CRUD para MongoDB

- Para realizar as operações no BD utilizaremos as funções da biblioteca do MongoDB:
 - insertOne → Inserir um novo documento.
 - replaceOne → Atualizar um documento existente.
 - find → Buscar documentos baseado nos filtros passados.
 - distinct → Análogo ao “SELECT DISTINCT”.
 - deleteOne → Excluir um documento.
- Documentação das funções da biblioteca:
<http://mongodb.github.io/node-mongodb-native/3.2/>

Operações no BD

- O formato dos métodos que farão interação com o BD ficará assim:

```
//Abrindo a conexão com o MongoDB
MongoClient.connect(uri, params, function (err, client) {
  if (!err) {
    var db = client.db(dbName); //Selecionando o banco para conectar
    db.collection(collName).FUNÇÃO(PARÂMETROS, CALLBACK () {
      if (!err) {
        //Sucesso na operação
      } else {
        //Erro na operação
      }
    });
  } else {
    //Erro na conexão com o MongoDB
  }
});
```

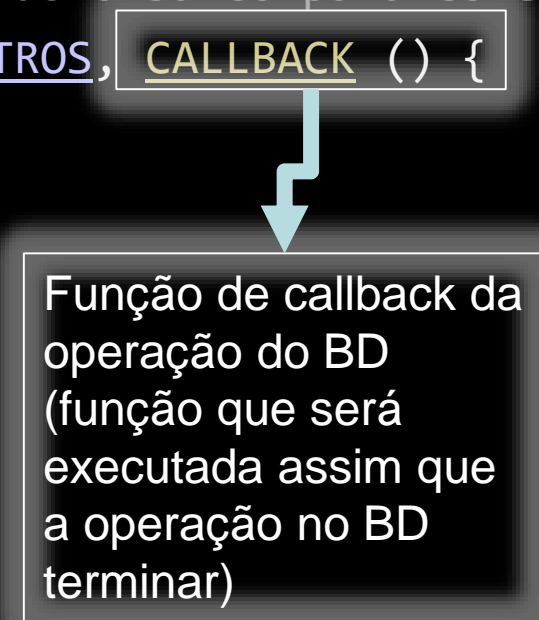


Parâmetros passados para operação.

Operações no BD

- O formato dos métodos que farão interação com o BD ficará assim:

```
//Abrindo a conexão com o MongoDB
MongoClient.connect(uri, params, function (err, client) {
  if (!err) {
    var db = client.db(dbName); //Selecionando o banco para conectar
    db.collection(collName).FUNÇÃO(PARÂMETROS, CALLBACK () {
      if (!err) {
        //Sucesso na operação
      } else {
        //Erro na operação
      }
    });
  } else {
    //Erro na conexão com o MongoDB
  }
});
```



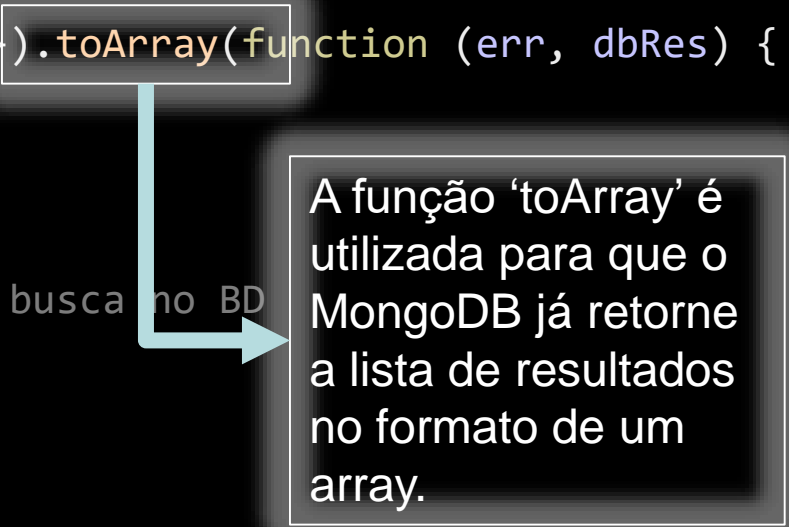
Função de callback da operação do BD (função que será executada assim que a operação no BD terminar)

Exemplo – Listar Todos

```
//Abrindo a conexão com o MongoDB
MongoClient.connect(uri, params, function (err, client) {
  if (!err) {
    var db = client.db(dbName); //Selecionando o banco para conectar
    //Executando busca no BD
    // .find({}) => filtro da busca
    // .project({}) => campos selecionados
    db.collection(collName).find({}).project({}).toArray(function (err, dbRes) {
      if (!err) {
        client.close();
        resolve(dbRes);
      } else {
        //Devolvemos um erro caso haja erro na busca no BD
        reject(err);
      }
    });
  } else {
    //Devolvemos um erro caso haja erro na conexão com o MongoDB
    reject(err);
  }
});
```

Exemplo – Listar Todos

```
//Abrindo a conexão com o MongoDB
MongoClient.connect(uri, params, function (err, client) {
  if (!err) {
    var db = client.db(dbName); //Selecionando o banco para conectar
    //Executando busca no BD
    // .find({}) => filtro da busca
    // .project({}) => campos selecionados
    db.collection(collName).find({}).project({}).toArray(function (err, dbRes) {
      if (!err) {
        client.close();
        resolve(dbRes);
      } else {
        //Devolvemos um erro caso haja erro na busca no BD
        reject(err);
      }
    });
  } else {
    //Devolvemos um erro caso haja erro na conexão com o MongoDB
    reject(err);
  }
});
```



A função 'toArray' é utilizada para que o MongoDB já retorne a lista de resultados no formato de um array.

Let's do it!

- Vamos refatorar o arquivo dao.js para que todas as operações de BD seja feita no MongoDB!

