Backend

The dark side

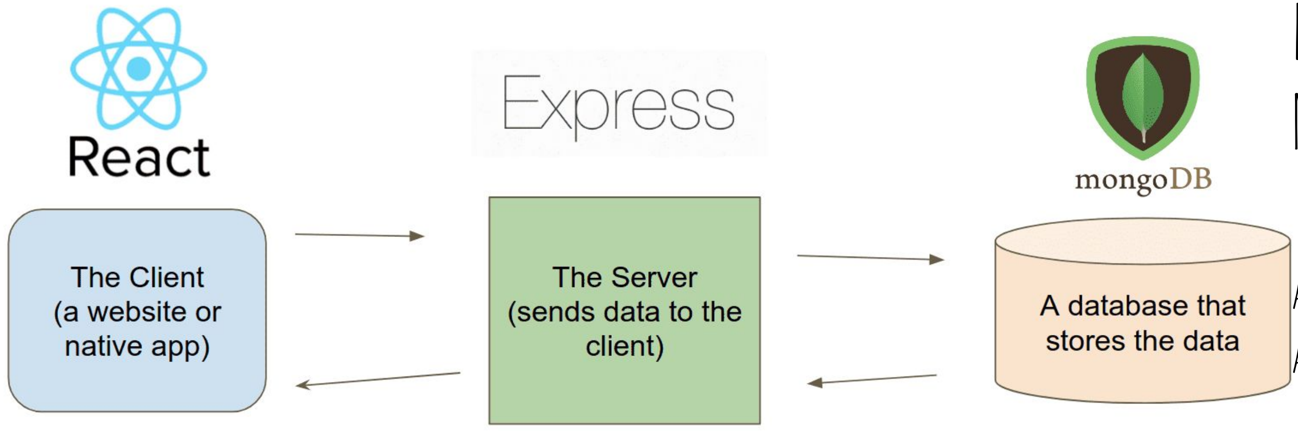
Remember The keeper app?

Big question? How was the data going to be sourced?

Where and how does that work?

MERN STACK

What infrastructure in a typical app looks like


React
Express
mongoDB
The Client (a website or native app)
The Server (sends data to the client)
A database that stores the data

This is running on your browser, computer, or phone

This is running in a datacenter (e.g. AWS, or Digital Ocean)

This is also running in a datacenter

M - MONGO DB
E - EXPRESS
R - REACT
N - NODE.JS

ALL THE ABOVE, REACT EXCLUDED ARE ALSO USED IN THE BACKEND

Database

This is an organized collection of data, generally stored and accessed electronically from a computer system.

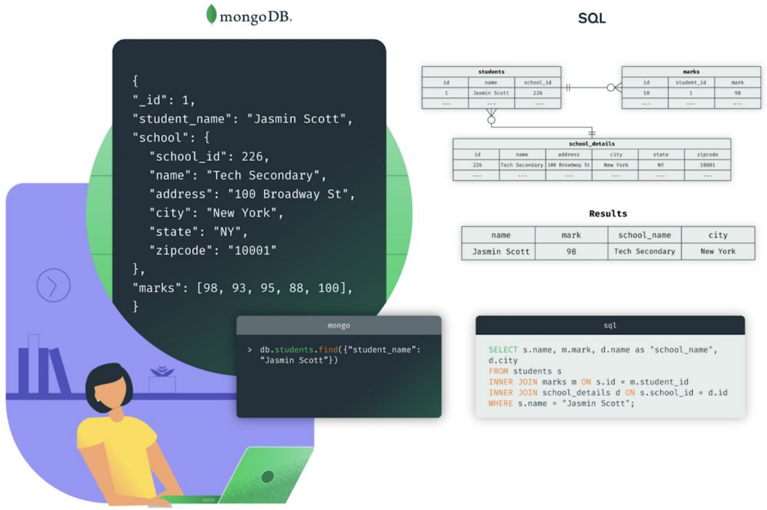
Types

Relational/ SQL - MySQL

nonRelational / NOSQL . - mongoDB

MONGO DB

MongoDB is a cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas.



```
{
"_id": 1,
"student_name": "Jasmin Scott",
"school": {
  "school_id": 226,
  "name": "Tech Secondary",
  "address": "100 Broadway St",
  "city": "New York",
  "state": "NY",
  "zipcode": "10001"
},
"marks": [98, 93, 95, 88, 100],
}
```

```
> db.students.find({"student_name":
  "Jasmin Scott"})
```

students

id	name	school_id
1	Jasmin Scott	226
...	...	...

marks

id	student_id	mark
10	1	98
...	...	...

school_details

id	name	address	city	state	zipcode
226	Tech Secondary	100 Broadway St	New York	NY	10001
...	...	...	...	...	...

Results

name	mark	school_name	city
Jasmin Scott	98	Tech Secondary	New York

```
SELECT s.name, m.mark, d.name as "school_name",
d.city
FROM students s
INNER JOIN marks m ON s.id = m.student_id
INNER JOIN school_details d ON s.school_id = d.id
WHERE s.name = "Jasmin Scott";
```

CRUD OPERATIONS

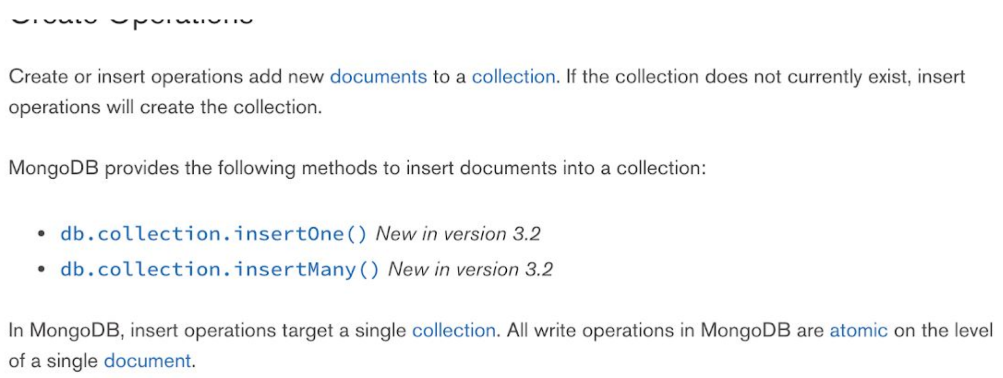
CRUD is an acronym that comes from the world of computer programming and refers to the four functions that are considered necessary to implement a persistent storage application: *create, read, update* and *delete*.

While using mongoDB these are just like any other API that connects you to your database. The Golden rule of APIs is it's completely ok to use them while reading the documentation. You don't have to memorize the syntax or functional requirements.

Mongodb document structure

```
{
   field1: value1,
   field2: value2,
   field3: value3,
   ...
   fieldN: valueN
}
```

CREATE OPERATION



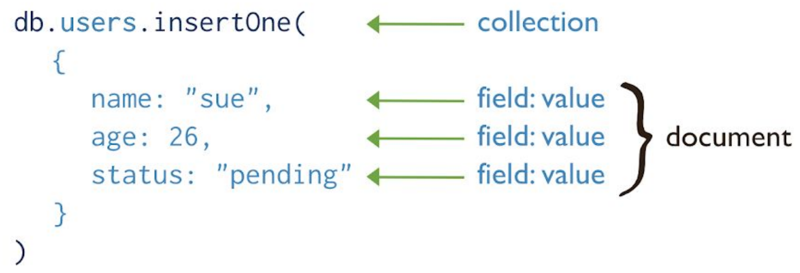
Create Operations

Create or insert operations add new documents to a collection. If the collection does not currently exist, insert operations will create the collection.

MongoDB provides the following methods to insert documents into a collection:

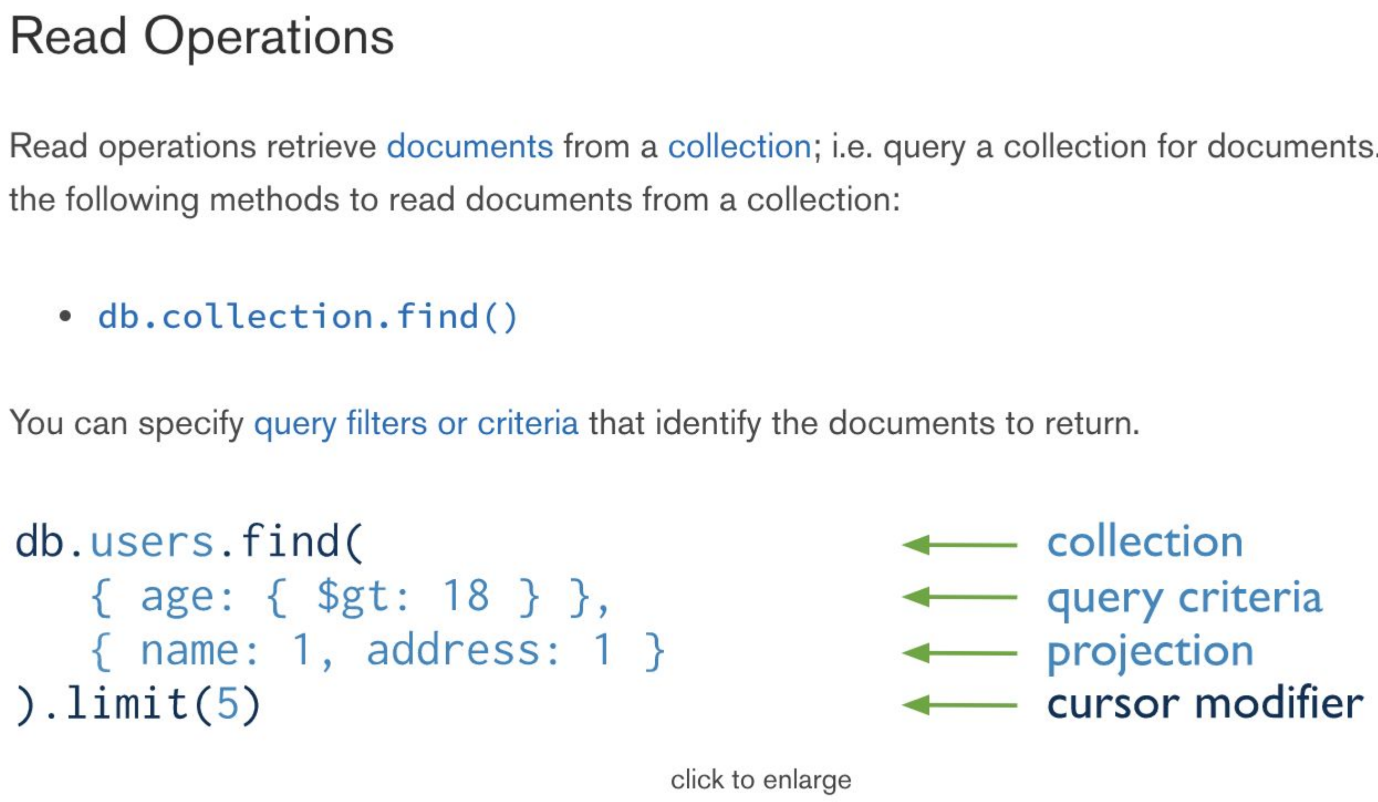
- `db.collection.insertOne()` *New in version 3.2*
- `db.collection.insertMany()` *New in version 3.2*

In MongoDB, insert operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.



```
db.users.insertOne(
   {
      name: "sue",
      age: 26,
      status: "pending"
   }
)
```

READ OPERATIONS



Read Operations

Read operations retrieve documents from a collection; i.e. query a collection for documents. MongoDB provides the following methods to read documents from a collection:

- `db.collection.find()`

You can specify query filters or criteria that identify the documents to return.

```
db.users.find(
   { age: { $gt: 18 } },
   { name: 1, address: 1 }
).limit(5)
```

click to enlarge

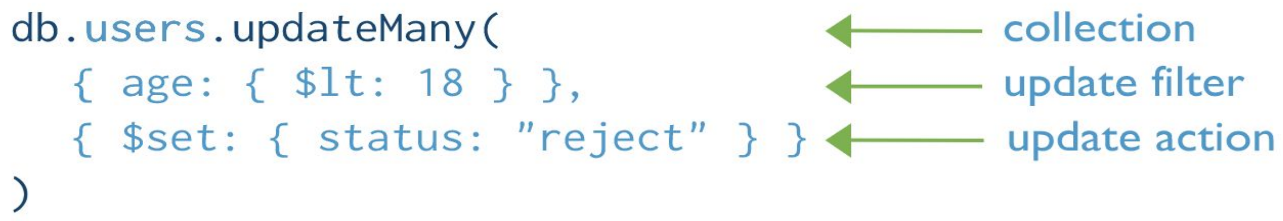
UPDATE OPERATIONS

Update operations modify existing documents in a collection. MongoDB provides the following methods to update documents of a collection:

- `db.collection.updateOne()` *New in version 3.2*
- `db.collection.updateMany()` *New in version 3.2*
- `db.collection.replaceOne()` *New in version 3.2*

In MongoDB, update operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

You can specify criteria, or filters, that identify the documents to update. These filters use the same syntax as read operations.



```
db.users.updateMany(
   { age: { $lt: 18 } },
   { $set: { status: "reject" } }
)
```

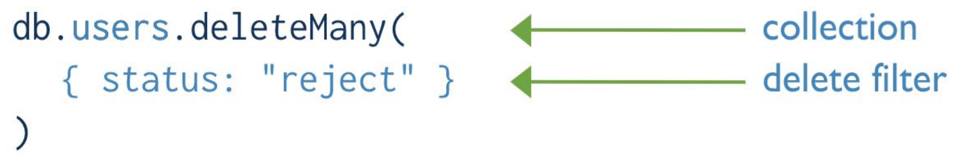
Delete operations

Delete operations remove documents from a collection. MongoDB provides the following methods to delete documents of a collection:

- `db.collection.deleteOne()` *New in version 3.2*
- `db.collection.deleteMany()` *New in version 3.2*

In MongoDB, delete operations target a single collection. All write operations in MongoDB are atomic on the level of a single document.

You can specify criteria, or filters, that identify the documents to remove. These filters use the same syntax as read operations.



```
db.users.deleteMany(
   { status: "reject" }
)
```

WHAT NEXTTTTT?

We just covered the basics of using MongoDB, but how does this relate to NODE.JS and our REACT APP???

Answer is: API. We need to build a restful API from our backend that our REACT App will use to source data displayed.

A few tools to know.

Express is a minimal and flexible Node.js web application framework that provides a robust set of features to develop web and mobile applications. It facilitates the rapid development of Node based Web applications. Following are some of the core features of Express framework −

- Allows to set up middlewares to respond to HTTP Requests.
- Defines a routing table which is used to perform different actions based on HTTP Method and URL.
- Allows to dynamically render HTML Pages based on passing arguments to templates.

MONGOOSE

When using node.js to code the backend and make the API, you need a way to connect, model and query the database.

The simple solution is using mongoose,a mongoDB object modeling tool designed to work in an asynchronous environment that provides a straight-forward, schema-based solution to model your application data.

ADDING DEPENDENCIES

It feels overwhelming mentioning all these technologies, however I can’t promise not to throw in four or five more modules.

This demonstrates the power of Node.js. Many third party developers are working round the clock to provide you with open access to libraries that makes work easier and faster.

rEVISITING REST APIS.