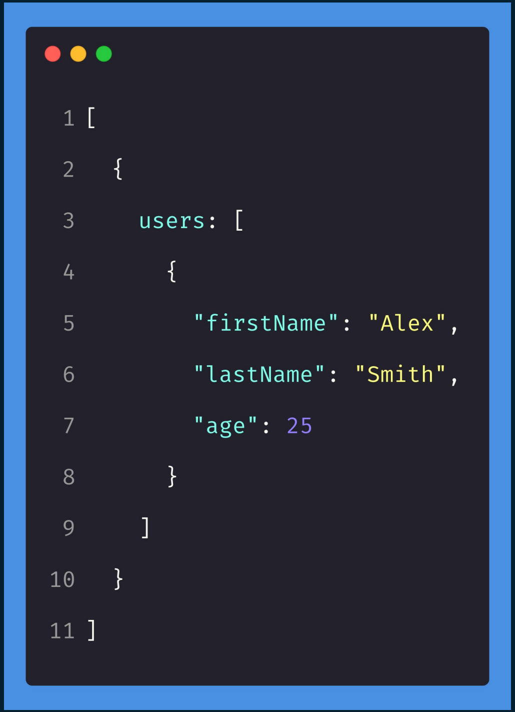
CODERSINHOODS

# Document-oriented database

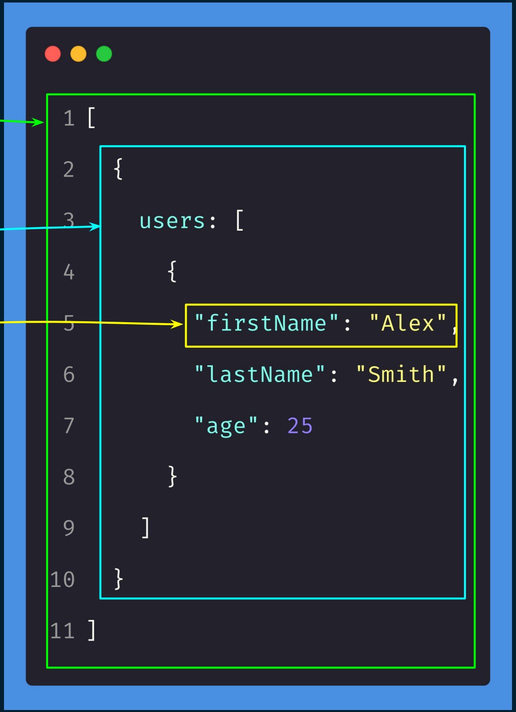
A **document database** is a type of non-relational database that is designed to store and query data as JSON-like documents.



```
[
  {
    users: [
      {
        "firstName": "Alex",
        "lastName": "Smith",
        "age": 25
      }
    ]
  }
]
```

# Vocabulary

| SQL | NoSQL |
|---|---|
| database | database |
| tables | collections |
| rows | documents |
| columns | fields |



```
[
  {
    users: [
      {
        "firstName": "Alex",
        "lastName": "Smith",
        "age": 25
      }
    ]
  }
]
```

Document databases are efficient and effective for storing catalog information. **For example**, in an e-commerce application, different products usually have different numbers of attributes. Managing thousands of attributes in relational databases is inefficient, and the reading performance is affected. Using a document database, each product's attributes can be described in a single document for easy management and faster reading speed. Changing the attributes of one product won't affect others.

## MongoDB

## Setup

1. Install *MongoDB* on your machine.
2. Create a DB folder
3. In your CLI run command to start your DB server

***Example:***

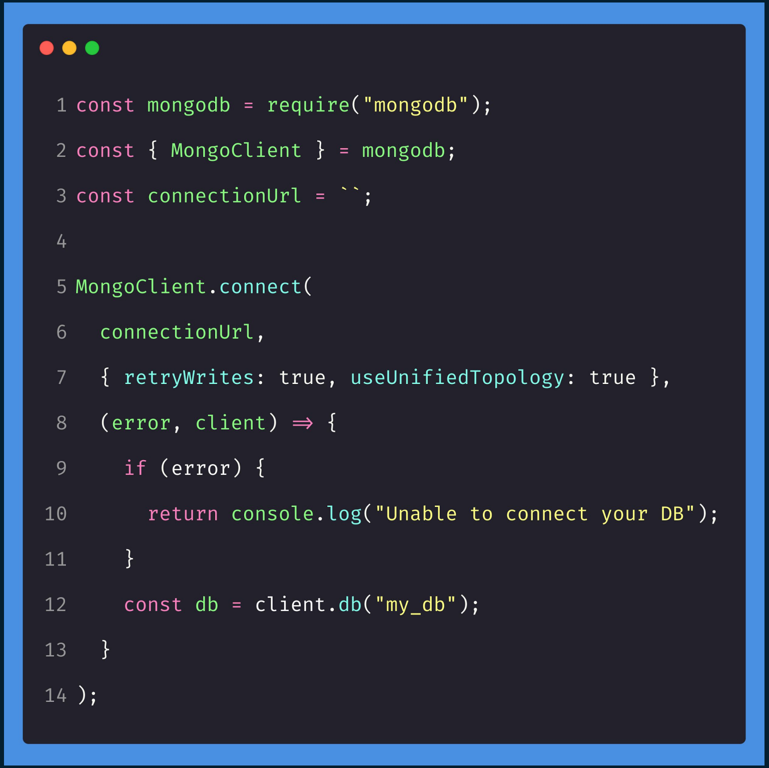
```
`/Users/vasile/mongodb/bin/mongod --dbpath=/Users/vasile/db`
```

**OR**

Use MongoDB cloud service

## Code setup

1. Install `mongodb` package
2. Import package.
3. Do basic setup



```
const mongodb = require("mongodb");
const { MongoClient } = mongodb;
const connectionUrl = ``;

MongoClient.connect(
  connectionUrl,
  { retryWrites: true, useUnifiedTopology: true },
  (error, client) => {
    if (error) {
      return console.log("Unable to connect your DB");
    }
    const db = client.db("my_db");
  }
);
```

**Documentation**
https://docs.mongodb.com/manual/introduction/

**CRUD tutorials**
https://docs.mongodb.com/guides/

## To code editor

CODERSINHOODS

## Free course

http://university.mongodb.com/

## Practice

Create API using mongoDB and express.

**End points:**

/doctors
`GET` - return all doctors
`POST` - add a new doctor

/patients
`GET` - return all patients
`POST` - add a new patient

/doctors/:id/patients
`GET` - return user by id

## To code editor

## Mongoose

https://mongoosejs.com/

*Mongoose provides a straight-forward, schema-based solution to model your application data. It includes built-in type casting, validation, query building, business logic hooks and more, out of the box.*