

inicio da aula

Criar uma conta no MongoDB Atlas

<https://www.mongodb.com/atlas/database>

Sign In

Don't have an account? [Sign Up](#)

Colocar todas as informações (Supergeeks)

Criar uma nova pasta "TerceiraAPI"

Entrar no terminal

```
>npm init -y
```

Entrar no VSCode c/ o code .

Abrir o terminal no VSCode e instalar o express

```
npm i express
```

No terminal, instalar o nodemon para atualizar nossa API automaticamente

```
npm i nodemon
```

No terminal, instalar o mongoose que são os pacotes para conexão com o banco de dados no MongoDB

```
npm i mongoose
```

Criar um script de start no package.json

```
{ } package.json X
{ } package.json > { } scripts > start
1  {
2    "name": "apiemongo",
3    "version": "1.0.0",
4    "description": "",
5    "main": "index.js",
6    "scripts": {
7      "test": "echo \"Error: no test specified\" && exit 1",
8      "start": "nodemon ./index.js"
9    },
10   "keywords": [],
11   "author": "",
12   "license": "ISC",
13   "dependencies": {
14     "express": "^4.17.3",
15     "mongoose": "^6.3.0",
16     "nodemon": "^2.0.15"
17   }
18 }
19
```

Criar o index.js na raiz do projeto

```
// Importar o pacote do express
const express = require('express')

// Inicializar o pacote
const server = express()

// Configurar a porta
server.listen(3000, () => {
  console.log("Servidor está funcionando...")
})

// Habilitar a leitura e recebimento de json // middlewares -> estabelece como será nossa comunicação com o banco.

server.use(
  express.urlencoded({
    extended: true,
  }),
)

server.use(express.json())

// Criar nossa rota inicial
server.get('/', (req, res) => {
  res.json({
    message: "Bem Vindo a nossa API com MongoDB"
  })
})
```

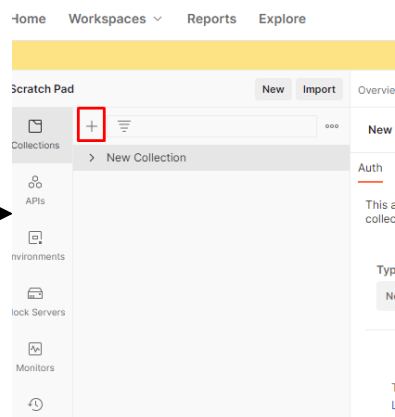
Iniciar o servidor com  
no terminal do  
VSCode

npm start

Baixar e instalar o  
Postman

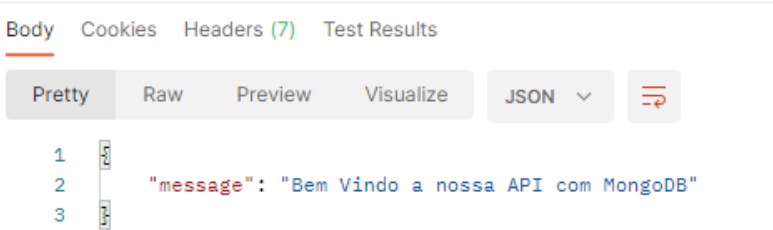
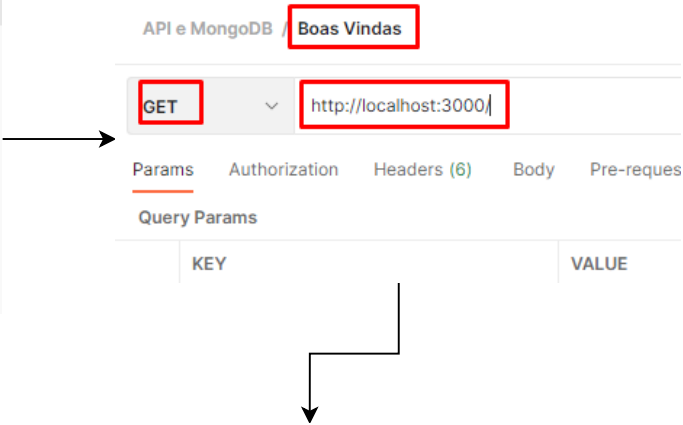
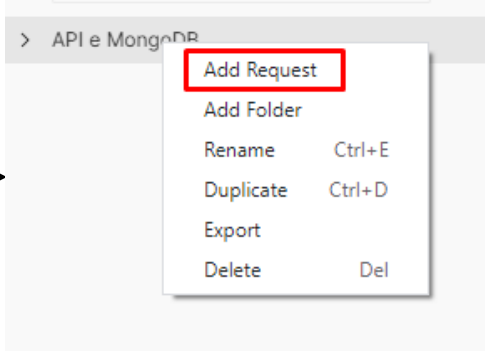
<https://www.postman.com/downloads/>

Criar uma nova  
collection, para  
organizar o projeto

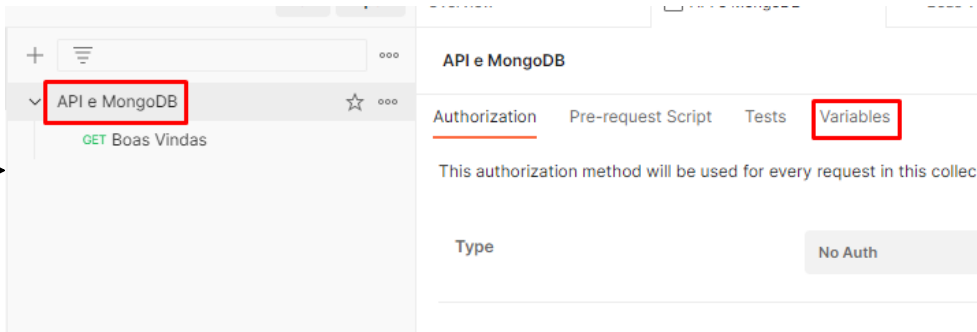


Atribuir o nome "API  
e MongoDB"

Dentro da collection, criar um request



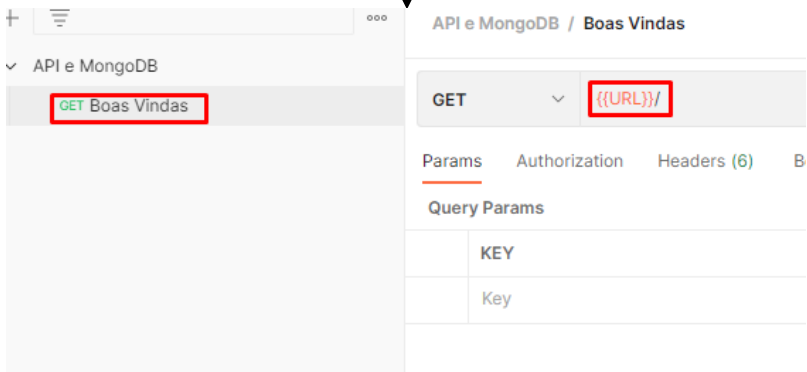
Criar uma variavel no postman para a URL



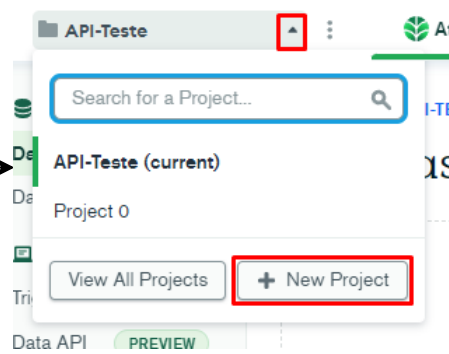
These variables are specific to this collection and its requests. [Learn more about collection variable](#)

	VARIABLE	INITIAL VALUE ⓘ	CURRENT VALUE ⓘ
<input checked="" type="checkbox"/>	URL	http://localhost:3000	http://localhost:3000
	Add a new variable		

Salvar



Entrar no mongo atlas e abrir um novo projeto



Dar um nome

## Create a Project

✓ Name Your Project Add Members

### Add Members and Set Permissions

Invite new or existing users via email address...

Give your members access permissions below.

dextervi@hotmail.com  
(you)

Project Owner

Cancel

← Go Back

Create Project




## Create a database

Choose your cloud provider, region, and specs.

Build a Database

Once your database is up and running, live migrate an existing MongoDB database into Atlas with our [Live Migration Service](#).

PREVIEW

 **Serverless**

For serverless applications that aren't critical with variable traffic. Minimal configuration required.

✓ Pay only for the operations you run


✓ Resources scale seamlessly to meet your workload

✓ Always-on security and backups

Create

Starting at  
**\$0.30/1M reads**

ADVANCED

 **Dedicated**

For production applications with sophisticated workload requirements. Advanced configuration controls.

✓ Network isolation and fine-grained access controls


✓ On-demand performance advice

✓ Multi-region and multi-cloud options available

Create

Starting at  
**\$0.08/hr\***  
\*estimated cost \$56.94/month

FREE

 **Shared**

For learning and exploring MongoDB in a cloud environment. Basic configuration options.

✓ No credit card required to start

✓ Explore with sample datasets

✓ Upgrade to dedicated clusters for full functionality

Create

Starting at  
**FREE**

## 1 How would you like to authenticate your connection?

Your first user will have permission to read and write any data in your project.

Username and Password

Certificate

Create a database user using a username and password. Users will be given the *read and write to any database* [privilege](#) by default. You can update these permissions and/or create additional users later. Ensure these credentials are different to your MongoDB Cloud username and password.

### Username

### Password

[Autogenerate Secure Password](#)[Copy](#)

Create User

## Add entries to your IP Access List

Only an IP address you add to your Access List will be able to connect to your project's clusters. You can manage existing IP entries via the [Network Access Page](#).

### IP Address

### Description

Add Entry


Add My Current IP Address

### IP Access List

### Description

177.27.250.231/32

My IP Address

 REMOVE

## Database Deployments

● **APICluster** **Connect** View Monitoring Browse Collections ...

● R 0  
● W 0  
Last 16 seconds  
100.0/s

● Connections 0  
Last 16 seconds  
100.0


VERSION	REGION	CLUSTER TIER	TYPE	BACKL
5.0.7	AWS / Sao Paulo (sa-east-1)	M0 Sandbox (General)	Replica Set - 3 nodes	Inact


Após configurar,  
realizar a ligação com  
o banco


✓ Setup connection security > Choose a connection method > Connect

Choose a connection method [View documentation](#)

Get your pre-formatted connection string by selecting your tool below.

**Connect with the MongoDB Shell**  
Interact with your cluster using MongoDB's interactive Javascript interface

**Connect your application**  
Connect your application to your cluster using MongoDB's native drivers

**Connect using MongoDB Compass**  
Explore, modify, and visualize your data with MongoDB's GUI

Go Back Close

## Connect to APICluster

✓ Setup connection security > ✓ Choose a connection method > Connect

1 Select your driver and version

DRIVER VERSION

Node.js 4.0 or later

2 Add your connection string into your application code

☐ Include full driver code example

```
mongodb+srv://vinicius:<password>@apicluster.eaftv.mongodb.net/myFirstDatabase?
retryWrites=true&w=majority
```

Replace **<password>** with the password for the **vinicius** user. Replace **myFirstDatabase** with the name of the database that connections will use by default. Ensure any option params are [URL encoded](#).

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back Close

No caminho copiado,  
substituir o password  
e o nome do banco

```
//mongodb+srv://vinicius:super123@apicluster.eaftv.mongodb.net/myFirstDatabase?retryWrites=true&w=majority
```

Conectar no código

```
//Importar o pacote do express
const express = require('express')

//Importar o pacote do mongoose
const mongoose = require('mongoose')

//Inicializar o pacote
const server = express()

//Configurar a porta
mongoose
  .connect('mongodb+srv://vinicius:super123@apicluster.eaftv.mongodb.net/bancodaapi?retryWrites=true&w=majority')
  .then(() => {
    server.listen(3000, () => {
      console.log("Servidor conectado ao MongoDB...")
    })
  })
  .catch((err) => {console.log(err)})

//Habilitar a leitura e recebimento de json // middlewares -> estabelece como será nossa comunicação com o banco.
```

Criar uma pasta na  
raiz do projeto  
chamada 'models'

São as entidades do  
nosso programa e do  
nosso banco

dentro da pasta  
models, criar um  
arquivo Movie.js

```
//Movie.js // mongoose
// Importar o pacote do mongoose
const mongoose = require('mongoose')

// Criar uma entidade de receba os metodos para criar, pegar, atualizar ... (cria uma tabela no banco de dados)
const Movie = mongoose.model('Movie', {
  name: String,
  year: Number,
  streaming: Boolean
})

// Exportar a entidade
module.exports = Movie
```

Importar o Movie no  
arquivo INDEX.js  
para utilizar

```
//Importar o pacote do mongoose
const mongoose = require('mongoose')

//Importar a entidade Movie
const Movie = require('./models/Movie')
```

no INDEX.js,  
construir a rota POST  
para salvar um dado  
no banco

```
//criar nossa rota para criar/salvar um filme no bando de dados -> POST
// async -> como leva um tempo e não temos garantia que vamos receber alguma informação, utilizamos essa função que retorna uma promessa.
server.post('/movie', async(req, res) => {
  //Pegar as informações do corpo da nossa requisição
  const {name, year, streaming} = req.body

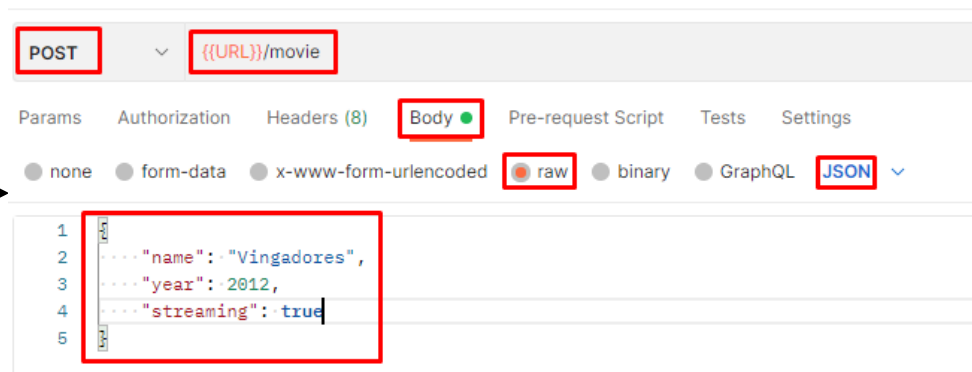
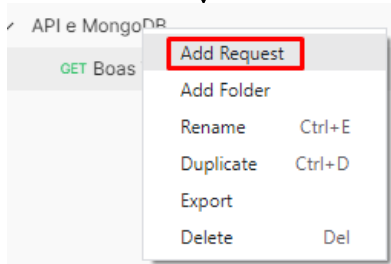
  //Validar as informações
  if (!name) {
    res.status(422).json({message: 'Nome é obrigatório'})
    return
  }

  //Colocar as informações em um objeto chamado movie
  const movie = {
    name,
    year,
    streaming
  }

  //criar/salvar essa informação no banco, mas pode falhar
  try{
    //esperar para garantir e criar os dados
    await Movie.create(movie)

    //enviar uma resposta com sucesso
    res.status(201).json({message: 'Filme inserido no banco com sucesso !'})
  } catch (error){
    //enviar uma resposta com o erro
    res.status(500).json({message: error})
  }
})
```

Testar no postman a  
nossa rota





## Database Deployments

● APICluster

Connect

View Monitoring

Browse Collections

...

● R 0  
● W

Connections 3.0

Last 2 minutes

## APICluster

Overview

Real Time

Metrics

Collections

Search

Profiler

Perfo

SANDBOX NODES REPLICASET

REGION Sao Paulo (sa-east-1)

● apiclu... shard-00-00.eaft...

SECONDARY

● apiclu... shard-00-01.eaft...

PRIMARY

● apiclu... shard-00-02.eaft...

SECONDARY

## This is a Shared Tier C

If you need a database that's better for high-performance p  
dedicated cluster.

Upgrade

+ Create Database

bancoapi

movies

## bancoapi.movies

STORAGE SIZE: 20KB TOTAL DOCUMENTS: 1 INDEXES TOTAL SIZE: 20KB

Find

Indexes

Schema Anti-Patterns 0

Aggregation

FILTER { field: 'value' }

## QUERY RESULTS 1-1 OF 1

```
_id: ObjectId("6262e94784805c355f979107")
name: "Vingadores"
year: 2012
streaming: true
__v: 0
```

no INDEX.js, construir  
uma rota GET para listar  
todos os filmes  
cadastrados

```
//Criar uma rota para listar todos os filmes cadastrados
server.get('/movies', async (req,res) => {

  // a busca pode falhar então é preciso tratar
  try{
    // metodo .find() lista todos os filmes do banco e coloca na variavel movies
    const movies = await Movie.find()

    //retorna em formato json
    res.status(200).json(movies)
  } catch (error){
    //retorna a mensagem de erro caso acontecer
    res.status(500).json({message: error})
  }
})
```

Resgatar os filmes no  
postman

Add Request

Add Folder

Rename Ctrl+E

Duplicate Ctrl+D

Export

Delete Del

API e MongoDB / Listar todos os filmes

GET {{URL}}/movies

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE
	Key	Value

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   {
3     "_id": "6262e94784805c355f979107",
4     "name": "Vingadores",
5     "year": 2012,
6     "streaming": true,
7     "__v": 0
8   },
9   {
10    "_id": "6262ed7ce0dfbc14a98d2659",
11    "name": "007 - Skyfall",
12    "year": 2008,
13    "streaming": true,
14    "__v": 0
15  }
16 }
```

no INDEX.js,  
construir uma rota  
para buscar somente  
um filme pelo id

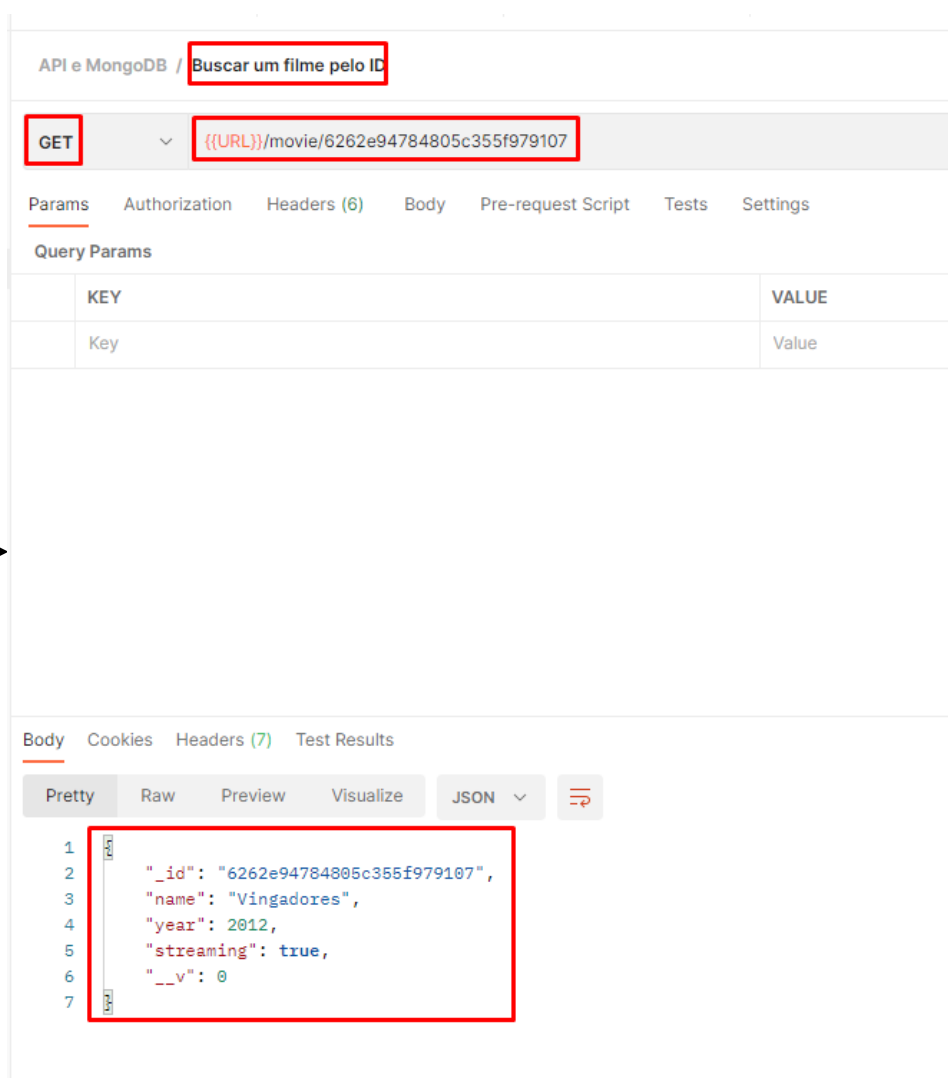
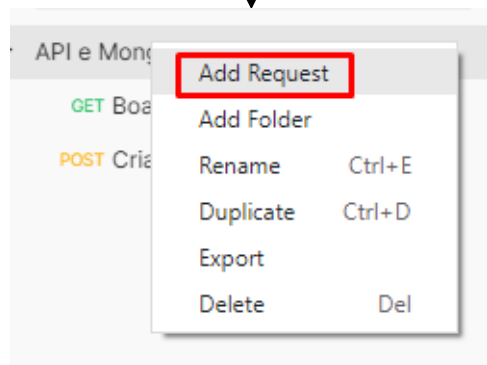
```
//Criar uma rota para buscar somente 1 filme pelo id
server.get('/movie/:id', async (req,res) => {
  const id = req.params.id

  try{
    //const movie = await Movie.findOne({_id: id})
    const movie = await Movie.findById(id)

    //validação caso não ache um usuário
    if (!movie) {
      req.status(422).json({message: "Nenhum filme encontrado"})
      return
    }

    res.status(200).json(movie)
  } catch (error) {
    res.status(500).json({message: error})
  }
})
```

Resgatar o filme no  
postman



no INDEX.js, atualizar os dados e um filme com PUT

```
//Atualizar os dados do usuário (PUT)
server.put('/movie/:id', async (req,res) => {
  const id = req.params.id

  const {name, year, streaming} = req.body

  const movie = {
    name,
    year,
    streaming
  }

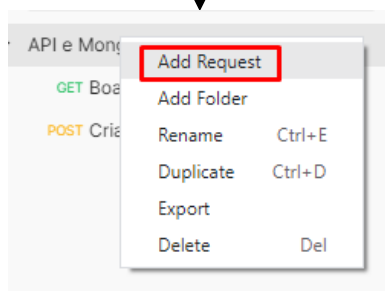
  try{
    const updatedMovie = await Movie.updateOne({_id: id}, movie)

    //Validação caso não encontre um filme
    if(updatedMovie.matchedCount === 0){
      res.status(422).json({message: "Filme não encontrado"})
      return
    }

    res.status(200).json(movie)

  } catch (error){
    res.status(500).json({message: error})
  }
})
```

criar a ação de atualizar no postman e testar



no INDEX.js, deletar  
um filme no banco

```
//Deletar um filme no banco de Dados (Delete)
server.delete('/movie/:id', async (req,res) => {
  const id = req.params.id

  try{
    const movie = await Movie.findById(id)
    await Movie.deleteOne({_id: id})
    res.status(200).json({message: "O filme foi removido com sucesso"})
  } catch (error){
    res.status(500).json({message: error})
  }
})
})
```

Adicionar a rota no  
postman

API e MongoDB / **Remover Filme**

**DELETE** **{{URL}}/movie/6262e94784805c355f979107**

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

	KEY	VALUE
	Key	Value

Bonus -> Rota para deletar tudo

no INDEX.JS, criar uma rota para deletar tudo

```
server.delete('/movies/deleteall', async(req,res) => {  
  try{  
    await Movie.deleteMany()  
    res.status(200).json({message: "Todos os filmes deletados"})  
  } catch (erro){  
    res.status(500).json({message: erro})  
  }  
})
```