



METCAMP

<WeB/>

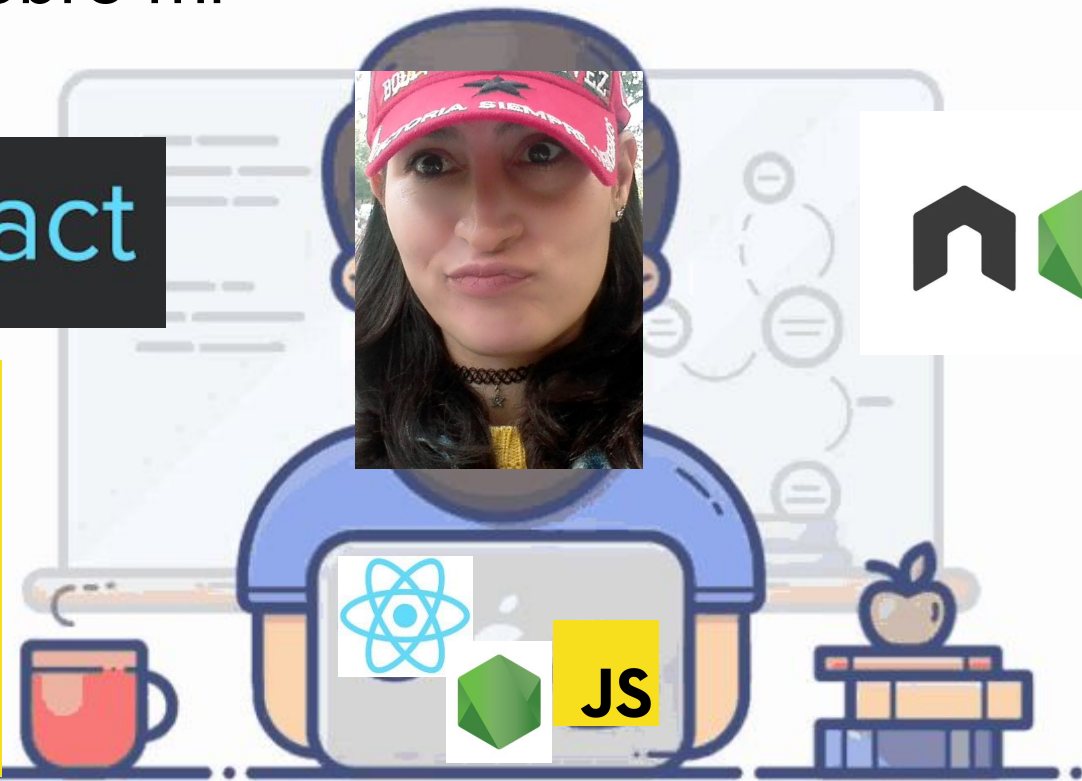
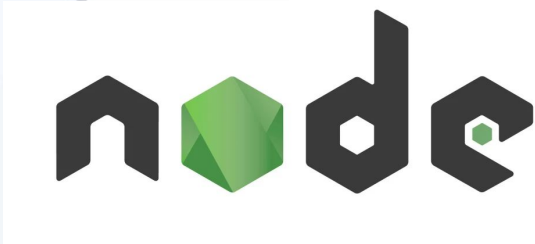
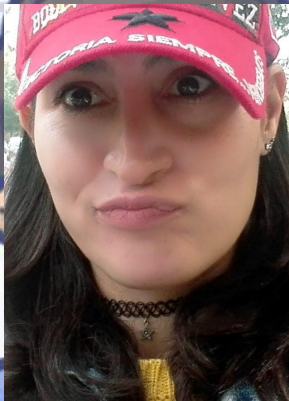
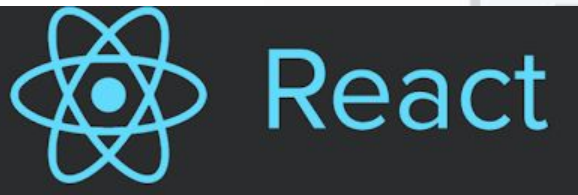
Base de datos

MongoDB

MET

[Mujeres en Tecnología]

Un poco sobre mi

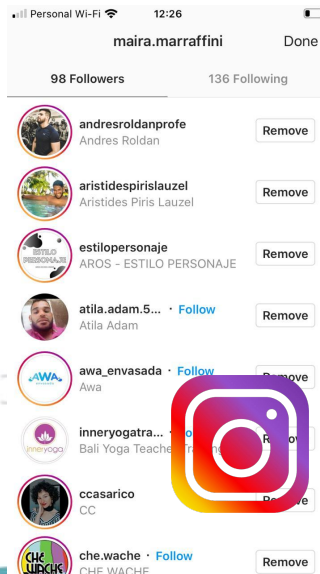


¿Qué vamos a estar viendo en el curso de hoy?

Base de datos



	A	B	C	D	E
1	Producto	Cantidad	Precio	Maximo Ganancia	
2	Coca Cola	100	2	200	
3	Pepsi	50	1,5	75	
4	Fanta	26	2	52	
5	Mirinda	56	3	168	
6	Sprite	203	2	406	
7					



En la cotidianidad ...

¿Qué nos permite guardar estas imágenes?



¿Y si recordamos un poco lo que vimos hasta ahora? #1

¿Qué sistema estamos creando? SISTEMA WEB DE PELÍCULAS

¿Qué partes conforman el sistema? frontend + backend

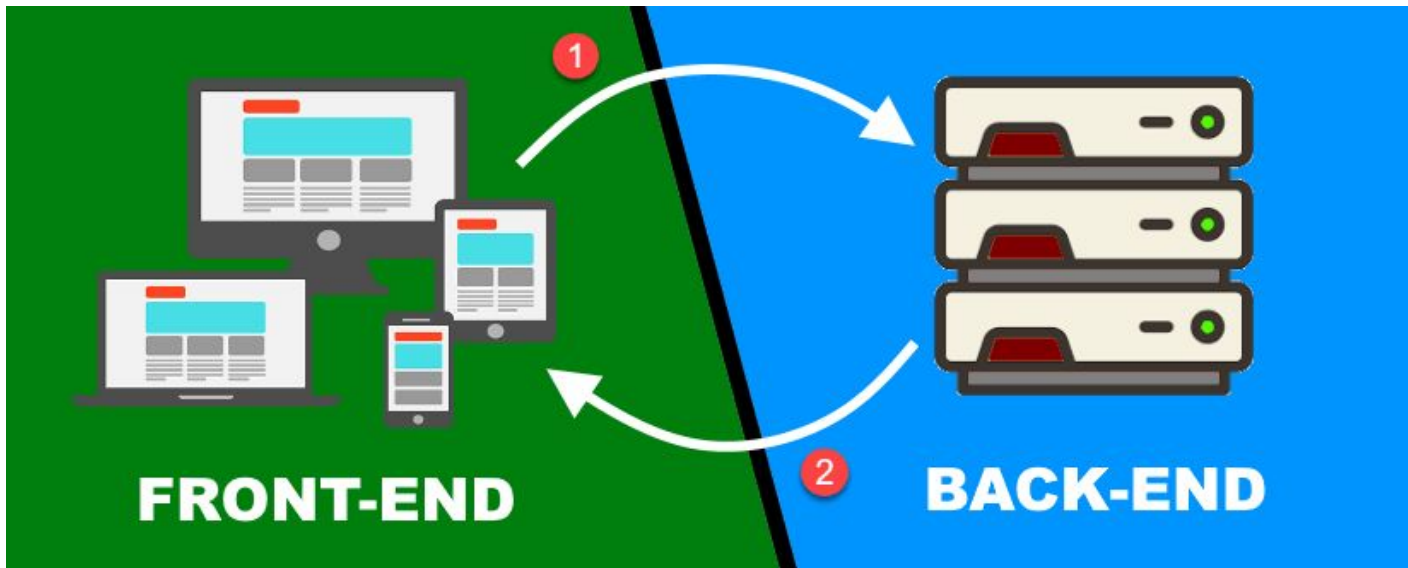
Frontend de ejemplo:

<https://pauburgos10.github.io/Met-pelis-metcamp/>

Backend de ejemplo:

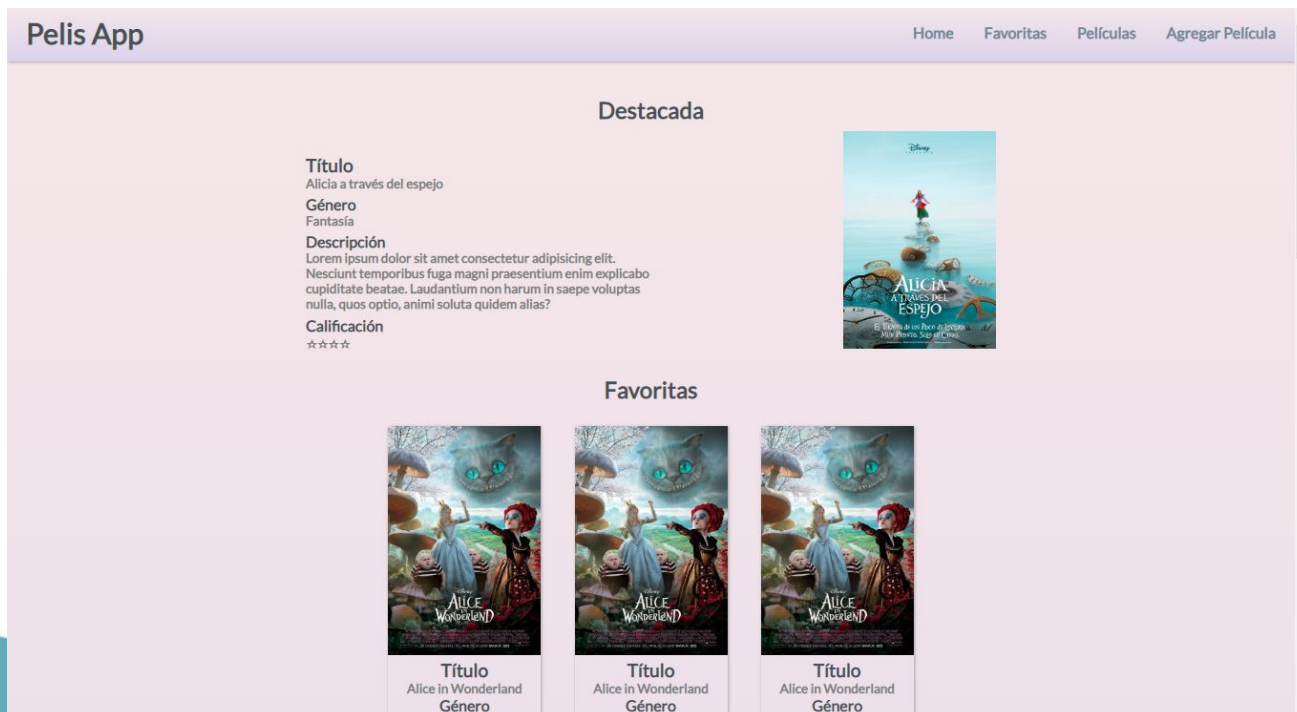
<http://localhost:3000/peliculas>

¿Y si recordamos un poco lo que vimos hasta ahora? #2



¿Y si recordamos un poco lo que vimos hasta ahora? #3

FRONTEND



¿Y si recordamos un poco lo que vimos hasta ahora? #3

FRONTEND

<https://github.com/pauburgos10/Met-pelis-metcamp/tree/master/docs>

index.html X

```
server > index.html > ...  
1 <!-- @format --  
2  
3 <!DOCTYPE html>  
4 <html lang="en"  
5 <head>  
6 <meta char=  
7 <meta name="viewport" content="width=  
8 <title>MeTCaMP</title>  
9 </head>  
10 <body>  
11 <div class="peliculas">  
12 <h2 class="mis-peliculas"></h2>  
13 </div>  
14 </body>  
15 </html>
```

HTML



style.css X

```
src > public > style.css > .seleted-mine  
21  
22 .field {  
23   height: 40px;  
24   width: 40px;  
25   font-size: 25px;  
26   text-align: center;  
27   font-family: 'Roboto Mono';  
28   border: 5px solid;  
29   box-sizing: border-box;  
30 }  
31  
32 .covered {  
33   border-color: ■ #ffffff ■ #808080 ■ #808080 ■ #ffffff;  
34 }
```

CSS



gameLogic.js X

```
src > public > gameLogic.js > GameLogic  
213  
214 function updateBoardWithFlags() {  
215   for (let i = 0; i < rows; i++)  
216     for (let j = 0; j < cols; j++)  
217       if (containsAMine(i, j)) {  
218         setLastGame(i, j, FLAGGED)  
219       }  
220 }
```

JS

¿Y si recordamos un poco lo que vimos hasta ahora? #4

Backend



Express
{API}

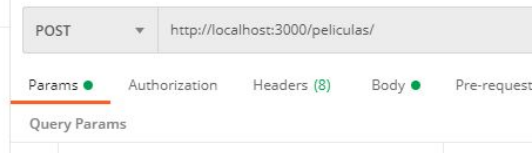


POSTMAN



mongoDB

```
app.listen(process.env.PORT, () => {  
  console.log(`server corriendo en http://localhost:3000`);  
});  
  
> app.get('/películas', func  
> app.get('/películas/:id',  
> app.post('/películas', fun  
> app.put('/películas/:id',  
> app.delete('/películas/:id'
```



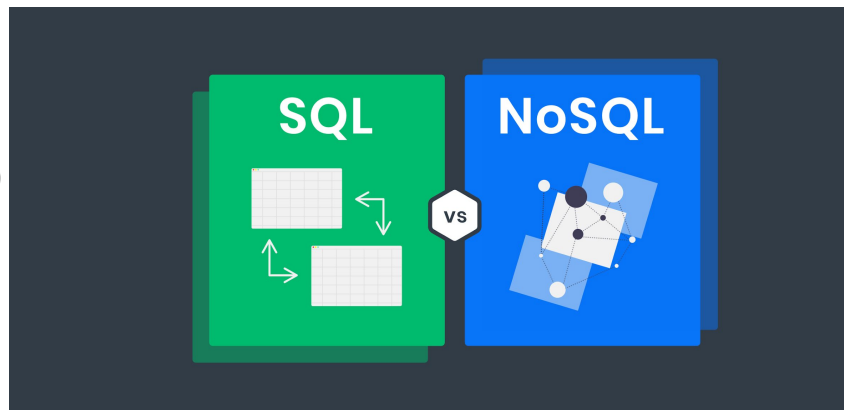
Un poco de teoría... solo un poco

¿Por Qué elegimos MongoDB?

- ★ es una base de datos NoSQL
- ★ es uno de los más elegidos en el mercado
- ★ es simple y fácil para comenzar

¿Qué es MongoDB?

- <https://es.wikipedia.org/wiki/MongoDB>



Manos a la obra...

Requisitos de instalación

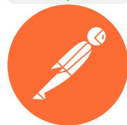
1. Visual Code Studio



2. Node.js



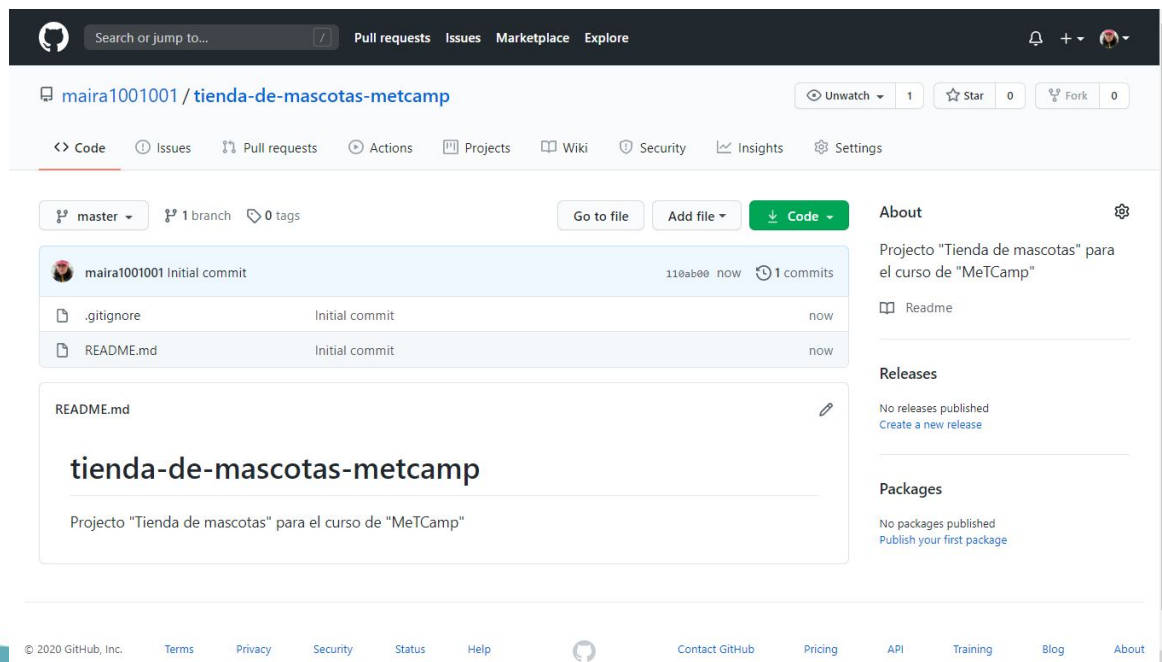
3. Postman



Manos a la obra...

Ahora entre todes

PASO 1: descargamos el sistema de películas desde github



Manos a la obra...

Por única vez

PASO 2: Instalamos Mongodb plugin en el VS Code

The screenshot shows the VS Code interface with the Extensions Marketplace open. The search bar contains 'mongodb'. The left sidebar lists several extensions, with 'MongoDB for VS Code' at the top. The right sidebar shows the details for the 'MongoDB for VS Code' extension by MongoDB. The extension is marked as a 'Preview' and has a rating of 4.5 stars. The description states: 'Connect to MongoDB and Atlas directly from your VS Code environment, navigate your databases ...'. The 'Install' button is visible. Below the description, there are tabs for 'Details', 'Feature Contributions', and 'Changelog'. The 'Details' tab is active, showing a 'PREVIEW' badge and a status message 'Azure Pipelines succeeded'. The 'Features' section lists: 'Navigate your MongoDB Data', 'Navigate your database, collections and read-only views', 'See the documents in your collections', and 'Get a quick overview of your schema'.

EXTENSIONS: MARKETPLACE

mongodb

MongoDB for VS Code 0.1.1 44K ★ 4.5
Connect to MongoDB and Atlas directly from your VS Code environmen...
MongoDB [Install](#)

Auto MongoDB 0.1.0 9K ★ 3
One click to run MongoDB without installation or configuration
Allen Lawrence [Install](#)

ES7 JavaScript/Node/Mongoose/MongoDB-MySQL snippets 0.4.0 8K ★ 5
Simple extension for Node, javascript, Mocha, Mysql, Mongodb, Mocha ...
abrahamwilliam007 [Install](#)

Azure Databases 0.15.0 281K ★ 3.5
Create, browse, and update globally distributed, multi-model databases...
Microsoft [Install](#)

Mongo php Snippets 1.0.3 1K ★ 5
snippets pack for work mongodb in php for vs code
hadi gholipor [Install](#)

Mongo Runner 0.6.2 33K ★ 3.5
MongoDB Runner is a VSCode extension to connect MongoDB instance.
Joey Yi Zhao [Install](#)

Azure Tools 0.0.11 205K ★ 4
Get web site hosting, SQL and MongoDB data, Docker Containers, Serve...
Microsoft [Install](#)

Mongo Snippets for Node.js 1.3.5 37K ★ 4.5
Provides snippets, boilerplate code for Mongo queries and completion s...
Rohan Mukherjee [Install](#)

Extension: MongoDB for VS Code

MongoDB for VS Code mongodb.mongodb-vscode [Preview](#)
MongoDB | 44.720 | ★★★★★ | Repository | License | v0.1.1
Connect to MongoDB and Atlas directly from your VS Code environment, navigate your databases ...
[Install](#)

[Details](#) [Feature Contributions](#) [Changelog](#)

MongoDB for VS Code [PREVIEW](#)

[Azure Pipelines](#) [succeeded](#)

MongoDB for VS Code makes it easy to work with MongoDB, whether your own instance or in [MongoDB Atlas](#).

Features

Navigate your MongoDB Data

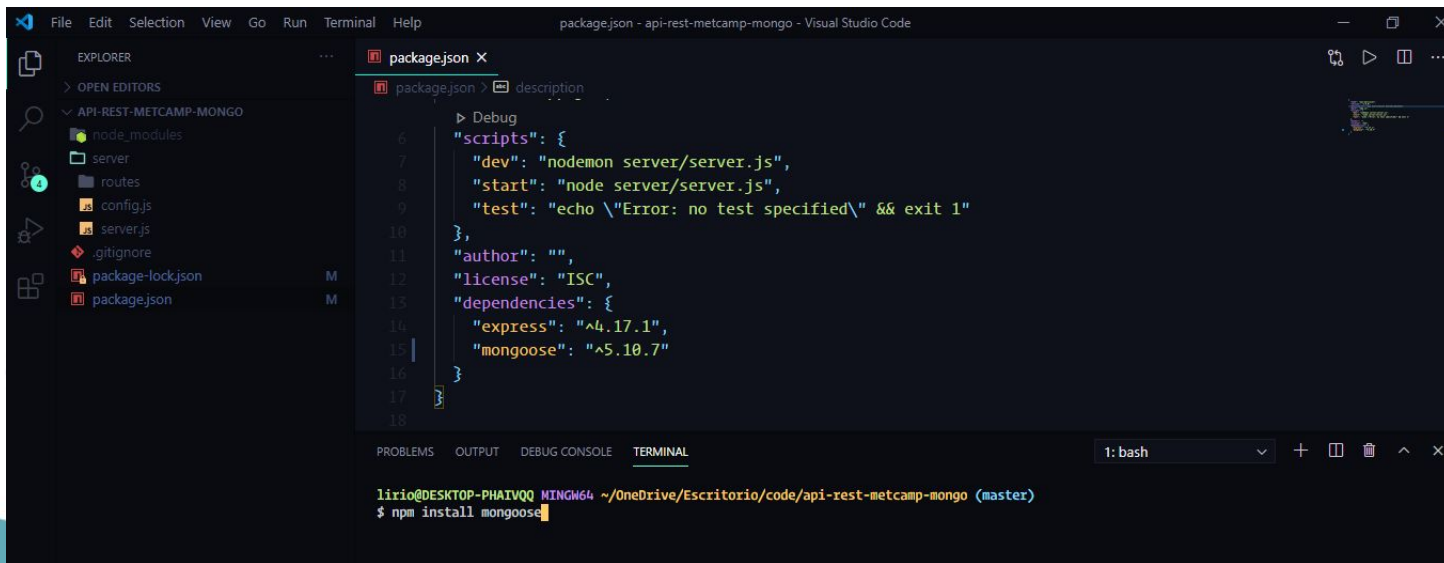
- Navigate your database, collections and read-only views
- See the documents in your collections
- Get a quick overview of your schema

Manos a la obra...

Ahora entre todes

PASO 3: : instalamos “mongoose”

- comando: `npm install mongoose`



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left shows the project structure for 'API-REST-METCAMP-MONGO', including files like 'server', 'routes', 'config.js', 'server.js', '.gitignore', 'package-lock.json', and 'package.json'. The main editor area displays the 'package.json' file with the following content:

```
6  > Debug
7  "scripts": {
8    "dev": "nodemon server/server.js",
9    "start": "node server/server.js",
10   "test": "echo \"Error: no test specified\" && exit 1"
11 },
12 "author": "",
13 "license": "ISC",
14 "dependencies": {
15   "express": "^4.17.1",
16   "mongoose": "^5.10.7"
17 }
18
```

At the bottom, the Terminal panel shows a bash shell with the command `npm install mongoose` entered.

Manos a la obra...

Ahora entre todes

PASO 4:

creamos la carpeta “models” y adentro
creamos el archivo “peliculas.js”

▼ METCAMP-API-PELICULAS

node_modules

server

models

pelicula.js

routes

peliculas.js

config.js

connect.js

server.js

.env

.gitignore

package-lock.json

package.json

README.md

Manos a la obra...

Ahora entre todes

PASO 5:

creamos el modelo de la
base de datos

<https://github.com/maira1001001/metcamels/pelicula.js>

```
PASO 2 Untitled-1 ● JS pelicula.js X
server > models > JS pelicula.js > ...
1  /** @format */
2
3  const mongoose = require('mongoose')
4  let Schema = mongoose.Schema;
5
6  let peliculasSchema = new Schema({
7    Titulo: String,
8    Genero: String,
9    Descripcion: String,
10   Calificacion: String,
11   imdbID: String,
12 });
13
14 module.exports = mongoose.model('Pel
```

Manos a la obra...

Ahora entre todes

PASO 6:

abrimos el archivo “server.js”,



pegamos código y

nos conectamos a la base de datos

```
server.js
server > . server.js > ...
1  require('./config')
2
3  const express = require('express')
4  const app = express()
5  const bodyParser = require('body-parser')
6  const mongoose = require('mongoose')
7
8  // parse application/x-www-form-urlencoded
9  app.use(bodyParser.urlencoded({ extended: false }))
10
11 // parse application/json
12 app.use(bodyParser.json())
13
14 app.use(require('./routes/peliculas'))
15
16
17
18 mongoose.connect('mongodb+srv://<user>:<password>@cluster0.rp7wm.mongodb.net/test1?retryWrites=
19 {useNewUrlParser: true, useUnifiedTopology: true, useNewUrlParser: false, useCreateIndex:
20 (err, res) => {
21   if (err) {
22     throw err
23   } else {
24     console.log('Base de datos ONLINE')
25   }
26 }
27
28 app.listen(process.env.PORT, () => {
29   console.log('server iniciado en puerto', process.env.PORT)
30 })
```

Manos a la obra...

PASO 7: Lo probamos con Postman

Untitled Request BUILD  

POST ▼ http://localhost:3000/mascotas/ Send Save ▼

Params ● Authorization Headers (8) Body ● Pre-request Script Tests Settings Cookies Codi

☐ none ☒ form-data ☐ x-www-form-urlencoded ☐ raw ☐ binary ☐ GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	nombre	michi			
<input checked="" type="checkbox"/>	tipo	gato			
	Key	Value	Description		

Manos a la obra...

Y ahora les toca a ustedes

Ejercicios prácticos #1

1. Registrar la película “Pretty Woman” usando Postman (Pista: usar “POST”)

```
{  
  "Titulo": "Pretty Woman",  
  "Genero": "Romántica",  
  "Descripcion": "Un elegante empresario hace el esmerado intento de adaptar a una mujer a las reglas de una sofisticada empresa de negocios",  
  "Calificacion": "2",  
  "imdbID": "tt0100405"  
}
```

2. Chequear desde VS Code que “Pretty Woman” se encuentre en la base de datos
3. Recuperar usando Postman la película “Pretty Woman” (Pista: usar “GET”)

Manos a la obra... Y ahora les toca a ustedes

Ejercicios prácticos #2

1. Registrar la película “Wonder Woman” usando Postman (Pista: usar “POST”)

```
{  
  "Titulo": "Wonder Woman",  
  "Genero": "Fantasía",  
  "Descripcion": "Es una película de superhéroes",  
  "Calificacion": "4",  
  "imdbID": "tt0451279"  
}
```

2. Chequear desde VS Code que “Woman Woman” se encuentre en la base de datos
3. Recuperar usando Postman la película “Wonder Woman” (Pista: usar “GET”)

Manos a la obra... Y ahora les toca a ustedes

Ejercicios prácticos #3

1. Registrar usando Postman, la película “The other Woman” (Pista: usar “POST”)

```
{  
  "Titulo": "The other Woman",  
  "Genero": "Comedia Romántica",  
  "Descripcion": "Tres mujeres unen fuerzas para vengarse de un hombre canalla, mentiroso e infiel que las traicionó a todas",  
  "Calificacion": "2",  
  "imdbID": "tt2203939"  
}
```

2. Chequear desde VS Code que “The other Woman” se encuentre en la base de datos
3. Recuperar usando Postman la película “The other Woman” (Pista: usar “GET”)

Manos a la obra... Y ahora les toca a ustedes

Ejercicios práctico usando POSTMAN

4. Recuperar la película **“Woman in Gold”** recién registrada (Pista: usar “GET”)
5. Actualizar el género de **“The other Woman”** a **“Fantasía”** (Pista: usar “PUT”)
6. Actualizar el ranking de **“Wonder Woman”** a **“3”** (Pista: usar “PUT”)
7. Obtener todas las películas registradas (Pista: usar “GET” sin parámetros)
8. Eliminar por completo la película **“Pretty Woman”** (Pista: usar “DELETE”)
9. Agregar la propiedad **“Director”** al modelo (Pista: modificar el archivo /models/peliculas.js)

Fin del curso

¡GRACIAS A TODES!

¡GRACIAS A LES ORGANIZADORES DEL METCAMP!

Twitter @MairaMarraffini

Email maira.marraffini@gmail.com