

# UNIVERSIDAD TECNOLÓGICA DE MÉXICO



**Nombre del alumno:** Jorge Andrés Bárcenas Cordero

**Matricula:** 315000169

**Nombre del docente:** Ing. Bandala

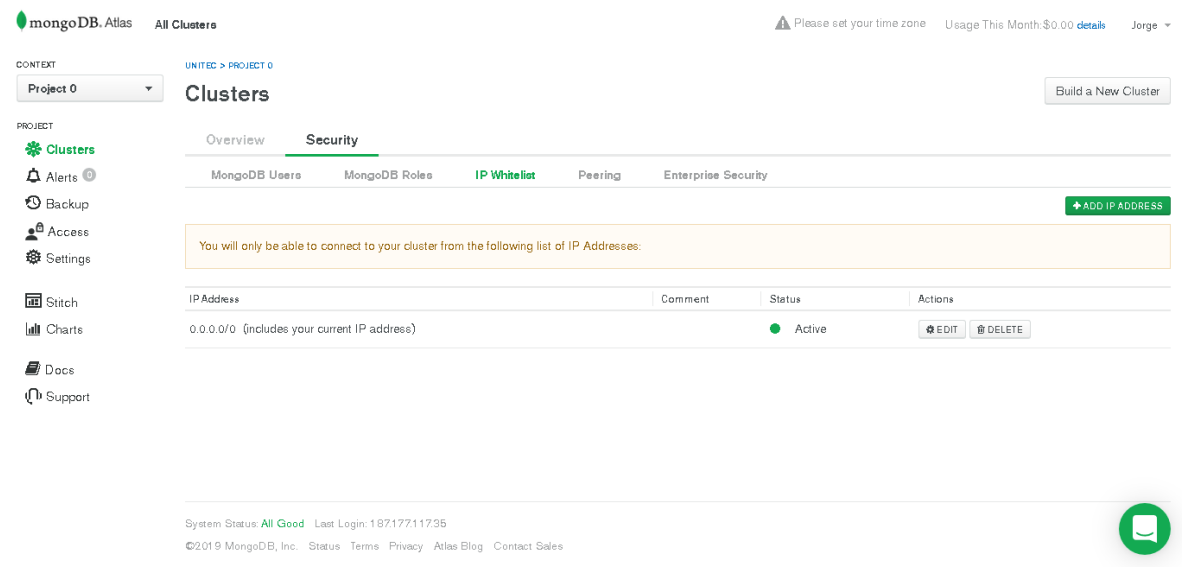
**Nombre de la materia:** Ciencia de Datos

**Entregable:** Documentación

**Fecha:** 26 abril 2019

# Direcciones IP

Se verifico, las direcciones IP, dentro de nuestra Cloud, donde se configura, cada una de las direcciones IP, para que se pueda acceder desde cualquier, dirección IP



The screenshot shows the MongoDB Atlas interface for a cluster named 'Project 0'. The 'Security' tab is selected, and the 'IP Whitelist' sub-tab is active. A message states: 'You will only be able to connect to your cluster from the following list of IP Addresses:'. Below this, a table lists the whitelisted IP addresses. The table has columns for 'IP Address', 'Comment', 'Status', and 'Actions'. One IP address is listed: '0.0.0.0/0 (includes your current IP address)', with a status of 'Active' and 'EDIT' and 'DELETE' action buttons.

IP Address	Comment	Status	Actions
0.0.0.0/0 (includes your current IP address)		Active	<a href="#">EDIT</a> <a href="#">DELETE</a>

# Comando NODE

Una vez configurado la conexión a la API, dentro de la aplicación Visual Studio Code, se abre una terminal, donde se ejecutará el comando, Node junto con el nombre del archivo que ejecutará **“APP.JS”**.



```
PS C:\Users\hnp\Desktop\MONGO-EXPRESS\MVC> node app.js
Server Inicializado en el puerto: 3000
Server mongodb Conectado...
```

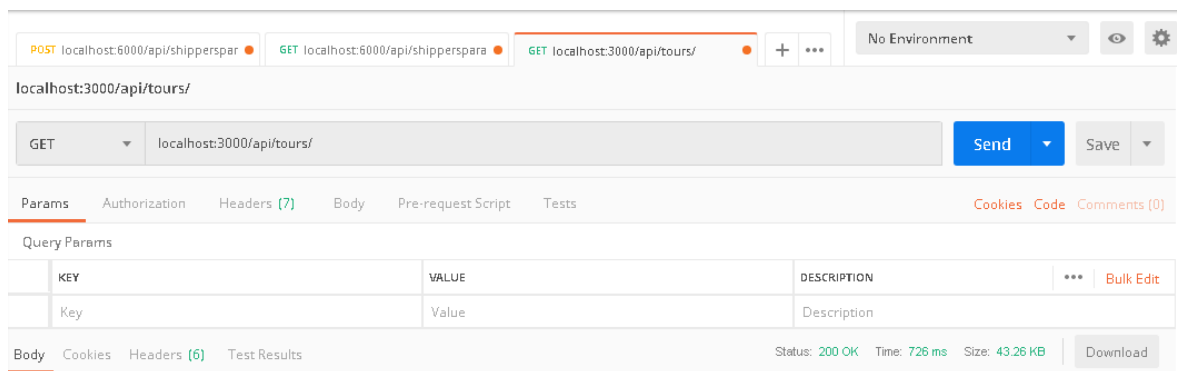
# Consulta Tour

Una vez ejecutado el comando, el sistema realizara conexión con el servidor.

```
PS C:\Users\hp\Desktop\MONGO-EXPRESS\MVC> node app.js
Server Inicializado en el puerto: 3000
Server mongodb Conectado...
```

Donde una vez conectado el servidor, se procede a abrir a aplicación Postman, en el cual dentro de la barra en la que se coloque, la URL. Donde se seleccionará la opción GET, para poder y se anexará la URL

**localhost:3000/api/tours/**



El cual consta del servidor local, el puerto por el que se realiza la petición, y el nombre de la solicitud, que se encuentra definido dentro del archivo **APP.JS**:

```
app.get('/', (req, res) => {
  res.send('Mongo Express.... Porfavor use /api/tours');
});
app.get('/api/tours', TourController.inq);
```

El cual nos dirige a el archivo amdmin.js, donde se encuentra, la especificación de la consulta que se realizara con la especificación enviada en la URL.

```
exports.inq = function (req, res) {
  Tour.find(function (err, tour) {
    console.log(tour);
    if (err) return console.log(err);
    res.send(tour);
    console.log("tours encontrados...");
  });
}
```

El cual toma la ayuda del archivo Tour.js, en el que con el uso del driver Mongoose, se crea un modelo para definir las propiedades de nuestro documento, almacenada en nuestra nube, esto para poder facilitar a búsqueda y obtención de información.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const tourModel = new Schema({
  tourName: { type: String, required: true },
  tourDifficulty: { type: String, required: false },
  tourDescription: { type: String, required: true },
  tourLength: { type: Number, required: true },
  tourPrice: { type: Number, required: true },
  tourTags: [String],
  tourOrganizer: {
    organizerName: { type: String, required: false },
    organizerPhone: { type: String, required: false }
  }
}, {collection: 'tour'});
module.exports = mongoose.model('Tour', tourModel);
```

Donde cabe mencionar, que se especifico la colección a la que se entrara para la obtención de datos, mediante el código:

```
{collection: 'tour'});
```

Ya que, al tener más de 1 colección en nuestra misma base de datos, a la hora de realizar la conexión a la base de datos, esta entra en conflicto, ya que, dentro del archivo Tour.js no se especifica la colección a la cual pertenece el modelo, creado para la obtención de información.

Donde una vez realizado el proceso, se dirige a la nube, junto a la base de datos y colecciona especificada, para posteriormente, obtener la información, la cual consta de la selección total, de la colección, **“db.tour.find()”**, por lo que la recogerla, la envía devuelta a el usuario para su despliegue.

GET

localhost:3000/api/tours/

Send

Save

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Cookies

Code

Comments (0)

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (6)

Test Results

Status: 200 OK

Time: 521 ms

Size: 43.26 KB

Download

Pretty

Raw

Preview

JSON

```
1 [
2   {
3     "tourOrganizer": {
4       "organizerName": "John Smith",
5       "organizerPhone": "555-555-5555"
6     },
7     "tourTags": [
8       "backpacking",
9       "Big Sur",
10      "hiking"
11    ],
12    "_id": "5cbf9d125476ba2a647aa252",
13    "tourBlurb": "Big Sur is big country. The Big Sur Retreat takes you to the most majestic part of the Pacific Coast and show you the secret trails.",
14    "tourName": "Big Sur Retreat",
15    "tourPackage": "Backpack Cal",
```

GET

localhost:3000/api/tours/

Send

Save

Params

Authorization

Headers (7)

Body

Pre-request Script

Tests

Cookies

Code

Comments (0)

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

Body

Cookies

Headers (6)

Test Results

Status: 200 OK

Time: 521 ms

Size: 43.26 KB

Download

Pretty

Raw

Preview

JSON

```
32     "sierra club"
33   ],
34   "_id": "5cbf9d125476ba2a647aa253",
35   "tourPackage": "Backpack Cal",
36   "tourBlurb": "\"Follow in the steps on John Muir, famous naturalist and founder of the Sierra Club, and walk the same trails he helped blaze in and around Yosemite National Park.\"\"",
37   "tourName": "In the Steps of John Muir",
38   "tourLength": 3,
39   "tourDescription": "This tour is not for the faint of heart though. This is a true backpacker's adventure, 37 miles in 3 days. Along the way, you'll bear witness to the classic monuments of Yosemite, such as Vernal Falls, Nevada Falls, Half Dome, Cathedral Peak, Tuolumne Meadows and Mt. Lyell. At Nevada Falls, the trail becomes narrow and from then on, you'll be privy to a \"secret\" Yosemite, including a rarely seen face of Half Dome. John Muir Trail tickets are required- and hard to come by- but Explore California has you covered. We'll also provide a checklist so that you come prepared with the right equipment and a seasoned guide for your tour.\"\"",
40   "tourRegion": "Northern California",
41   "tourPrice": 600,
```

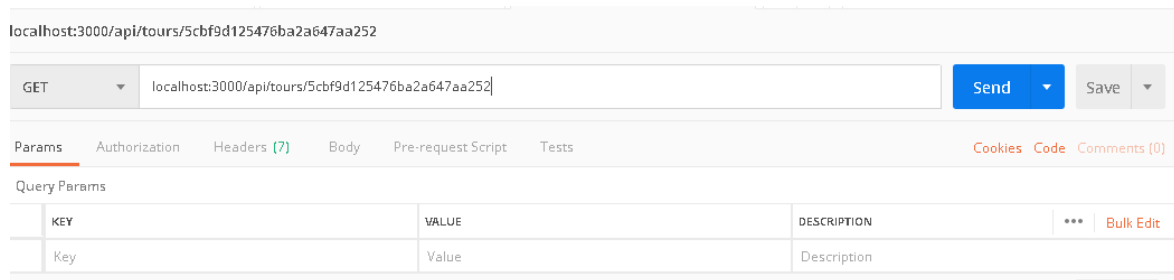
Y así observar toda la información desplegada de la búsqueda de la API, a la que se consulto mediante la URL ingresada.

## Consulta inqId

Se realizará una consulta a la colección Tour, dentro de nuestra base de datos en la nube, en la que constará de una búsqueda específica, por ID, en la que una vez posicionado en la aplicación Postman, se introducirá, en la barra de búsqueda URL, lo siguiente:

**localhost:3000/api/tours/"ID\_definido"**

En el que se ingresará algún ID, existente dentro de nuestra colección Tour, donde en este caso se hará uso del siguiente ID: **5cbf9d125476ba2a647aa252**.



El cual realizará el mismo proceso que la **consulta a la colección Tour**, previamente detallado, el cual inicia en el archivo app.js, donde busca, el formato al que pertenece.

```
app.get('/api/tours/:id', TourController.inqId);
```

El cual nos dirige al archivo admin.js donde busca, las especificaciones de la consulta que se realizaran, así como la leyenda que se desplegará en caso de tener éxito en la búsqueda de la información.

```
exports.inqByName = (req, res) => {
  Tour.findOne({tourName:req.params.name}, (err, tour) => {
    if (err) return console.log(err);
    res.send(tour);
    console.log("tour encontrado:"+req.params.name);
  })
};
```

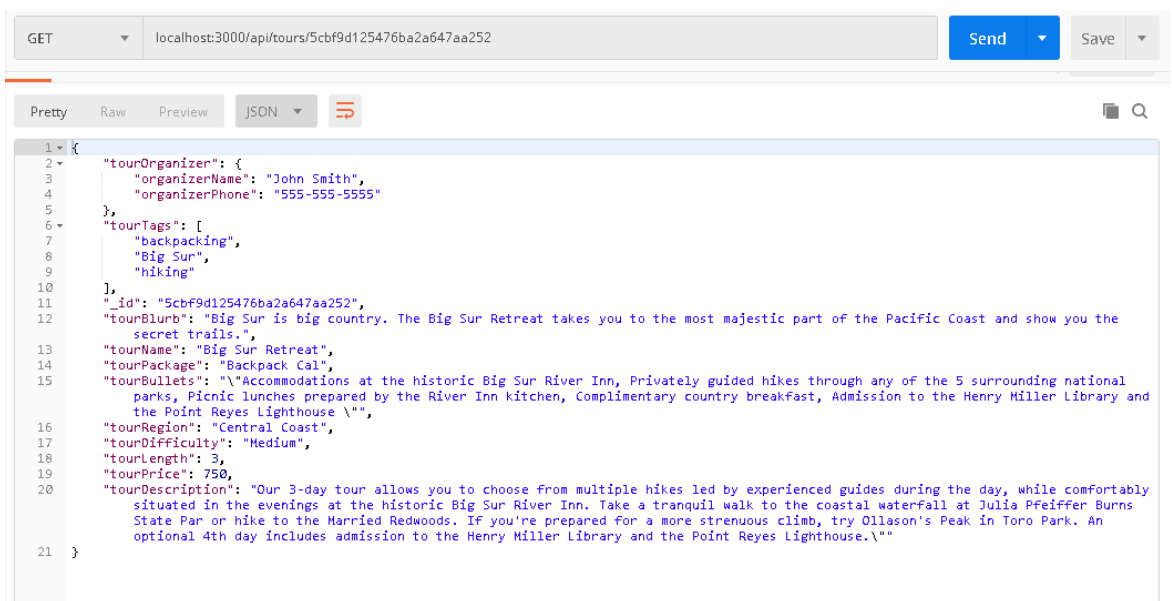
Donde posteriormente nos dirige, al archivo Tour.js, donde ya este especificado el modelo a usar, así como el nombre de la colección a la que se accederá.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const tourModel = new Schema({
  tourName: { type: String, required: true },
  tourDifficulty: { type: String, required: false },
  tourDescription: { type: String, required: true },
  tourLength: { type: Number, required: true },
  tourPrice: { type: Number, required: true },
  tourTags: [String],
  tourOrganizer: {
    organizerName: { type: String, required: false },
    organizerPhone: { type: String, required: false }
  }
}, {collection: 'tour'});
module.exports = mongoose.model('Tour', tourModel);
```

Donde posterior, se dirigirá a la base de datos dentro de nuestra nube, en la que se realizará la búsqueda de la información de la consulta, la cual corresponde a una consulta, de un registro (ID) específico:

**db.tour.find({ID: 5cbf9d125476ba2a647aa252})**

Donde una vez realizado el proceso, se obtendrá la información y se le desplegará al usuario en la aplicación Postman.

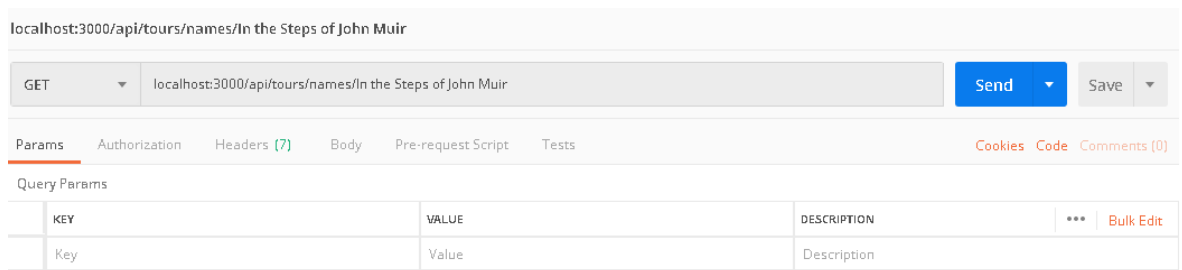


# Consulta inqByName

Se realizará una consulta dentro de la colección Tour, dentro de nuestra base de datos en Atlas, en la que constará de una búsqueda específica, por nombre del tour, en la que una vez posicionado en la aplicación Postman, se introducirá, en la barra de búsqueda URL, lo siguiente:

**localhost:3000/api/tours/" Nombre de tour definido"**

En el que se ingresará un Nombre de tour, existente dentro de nuestra colección Tour, para este ejemplo, se hará uso del siguiente nombre de tour: **In the Steps of John Muir**



The screenshot shows the Postman interface for a GET request. The URL bar contains 'localhost:3000/api/tours/names/In the Steps of John Muir'. Below the URL bar, there are tabs for 'Params', 'Authorization', 'Headers [7]', 'Body', 'Pre-request Script', and 'Tests'. The 'Params' tab is selected, showing a table with columns 'KEY', 'VALUE', and 'DESCRIPTION'. The table has one row with 'Key' in the KEY column, 'Value' in the VALUE column, and 'Description' in the DESCRIPTION column. There are also buttons for 'Send', 'Save', 'Cookies', 'Code', and 'Comments (0)'.

KEY	VALUE	DESCRIPTION
Key	Value	Description

De igual manera realizará el mismo proceso que las consultas previamente expuestas, donde una vez realizado la petición en Postman, inicia en el archivo app.js, donde busca, el formato al que pertenece.

```
app.get('/api/tours/names/:name', TourController.inqByName);
```

El cual redirige la información al archivo admin.js donde busca, las especificaciones de la consulta que se realizaran, así como la leyenda que se desplegará en caso de tener éxito en la búsqueda de la información.

```
exports.inqByName = (req, res) => {
  Tour.findOne({tourName:req.params.name}, (err, tour) => {
    if (err) return console.log(err);
    res.send(tour);
    console.log("tour encontrado:"+req.params.name);
  })
};
```



El cual solicita el archivo Tour.js, para basarse en el modelo, descrito en el archivo, para la búsqueda de las solicitudes, así como la colección a la que tendrá que acceder dentro de la base de datos.

```
const mongoose = require('mongoose');
const Schema = mongoose.Schema;
const tourModel = new Schema({
  tourName: { type: String, required: true },
  tourDifficulty: { type: String, required: false },
  tourDescription: { type: String, required: true },
  tourLength: { type: Number, required: true },
  tourPrice: { type: Number, required: true },
  tourTags: [String],
  tourOrganizer: {
    organizerName: { type: String, required: false },
    organizerPhone: { type: String, required: false }
  }
}, {collection: 'tour'});
module.exports = mongoose.model('Tour', tourModel);
```

Donde una vez realizado, el procedimiento se realizará la petición a nuestra base de datos en Atlas, para la obtención de información de la colección Tour, en la cual, se realizará, una consulta:

**db.tour.find({tourName:In the Steps of John Muir})**

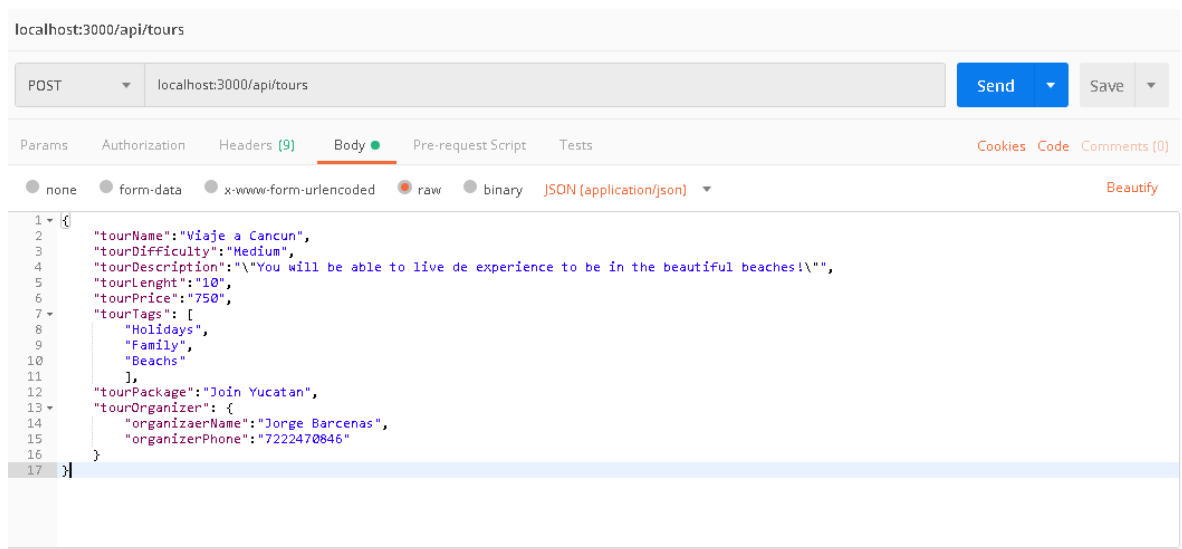
En el cual, al finalizar, se desplegará toda la información que cumpla con la condición, solicitada por el usuario dentro de la aplicación Postman.

```
{
  "tourOrganizer": {
    "organizerName": "John Smith",
    "organizerPhone": "555-555-5555"
  },
  "tourTags": [
    "backpacking",
    "john muir",
    "hiking",
    "sierra club"
  ],
  "_id": "5cbf9d125476ba2a647aa253",
  "tourPackage": "Backpack Cal",
  "tourBlurb": "\Follow in the steps on John Muir, famous naturalist and founder of the Sierra Club, and walk the same trails he helped blaze in and around Yosemite National Park.\",",
  "tourName": "In the Steps of John Muir",
  "tourLength": 3,
  "tourDescription": "This tour is not for the faint of heart though. This is a true backpacker's adventure, 37 miles in 3 days. Along the way, you'll bear witness to the classic monuments of Yosemite, such as Vernal Falls, Nevada Falls, Half Dome, Cathedral Peak, Tuolumne Meadows and Mt. Lyell. At Nevada Falls, the trail becomes narrow and from then on, you'll be privy to a \"secret\" Yosemite, including a rarely seen face of Half Dome. John Muir Trail tickets are required- and hard to come by- but Explore California has you covered. We'll also provide a checklist so that you come prepared with the right equipment and a seasoned guide for your tour.\",",
  "tourRegion": "Northern California",
  "tourPrice": 600,
  "tourDifficulty": "Difficult",
  "tourBullets": "\Guided hike of the first 37 miles of the John Muir Trail, Reserved campsites, Organic and unexpectedly delicious prepared trail meals, Overview of Yosemite's history and iconic vistas, Wilderness survival lessons \""
```

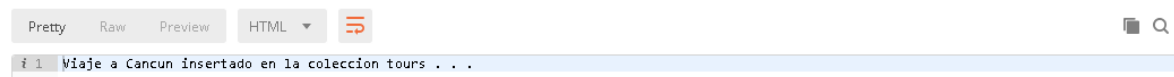
# Insertar Tour

Esta ocasión se realizará un insert dentro de la colección Tour dentro de nuestra base de datos learning\_mongo, almacenada en nuestra nube Atlas, en la cual en el apartado izquierdo donde se coloca la URL, se tendrá que cambiar la condición de **GET a POST**.

De igual manera, se tendrá que dirigir a la pestaña “Body”, posteriormente seleccionar la opción “Raw”, en la cual nos desplegará una terminal, donde se tendrá que introducir, toda a información que se requiera registrar en la colección.



Donde se nos desplegará una confirmación en la terminal, cuando se allá generado la información dentro de la colección.



La cual, dicha leyenda esta especificada, dentro del archivo Admin.js

```
tour.save(function (err, tour) {
  if (err) return console.error(err);
  res.send(tour.tourName + " insertado en la coleccion tours . . .");
  //console.log("Tour insertado " +req.params.name);
  console.log("Tour insertado: " +tour.tourName);
});
```

En el cual podemos verificar, la información cuando se realiza una respectiva consulta dentro de la colección Tour. Verificando de esta manera, el almacenamiento de la información previamente enviada.

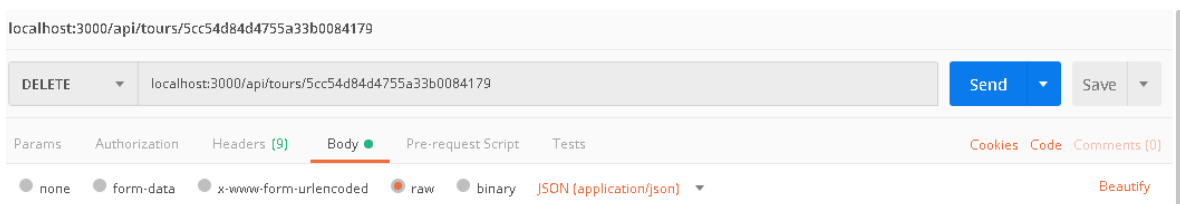


## Eliminar Tour

Se realizará la eliminación completa de un objeto, para este servicio, se requiere posicionarse dentro la aplicación Postman, en la que se requiere seleccionar el servicio DELETE, de la parte izquierda donde se coloca la URL.

Y escribir sobre la parte de URL, la siguiente línea:

**localhost:3000/api/tours/5cc54d84d4755a33b0084179**



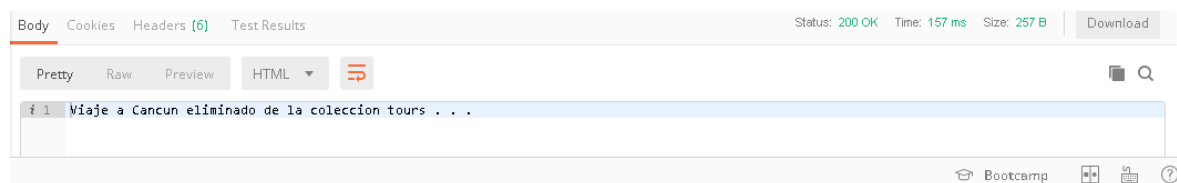
Una vez introducido la información correspondiente, el programa, inicia en el archivo Admi.js, en el que obtiene el nombre de la operación que realizamos.

```
app.delete('/api/tours/:id', TourController.delete);
```

Para posterior, dirigirse al archivo Admin.js, en el que se encuentra definida la operación a realizar dentro de la base de datos.

```
exports.delete = (req, res) => {
  Tour.findByIdAndDelete(req.params.id, (err, tour) => {
    if (err) return console.log(err);
    res.send(tour.tourName + " eliminado de la coleccion tours . . .");
    console.log("tour eliminado " + req.params.id);
  })
};
```

Donde procede a acceder a la base de datos, para ejecutar las instrucciones, donde le envía una respuesta a el usuario mediante una leyenda, confirmándole al usuario que la instrucción enviada resulto exitosa.



Donde si se realiza una consulta del elemento eliminado, no se desplegará ninguna información.

