

ESCUELA SUPERIOR DE INGENIERÍA

Programación en Internet

Grado en Ingeniería Informática

Servicios Web & jQuery

Autores:

José Manuel Periñán Freire

Iñaki Urrutia Sanchez

Supervisores:

David Corral Plaza

Cádiz, 26 de 11 de 2018

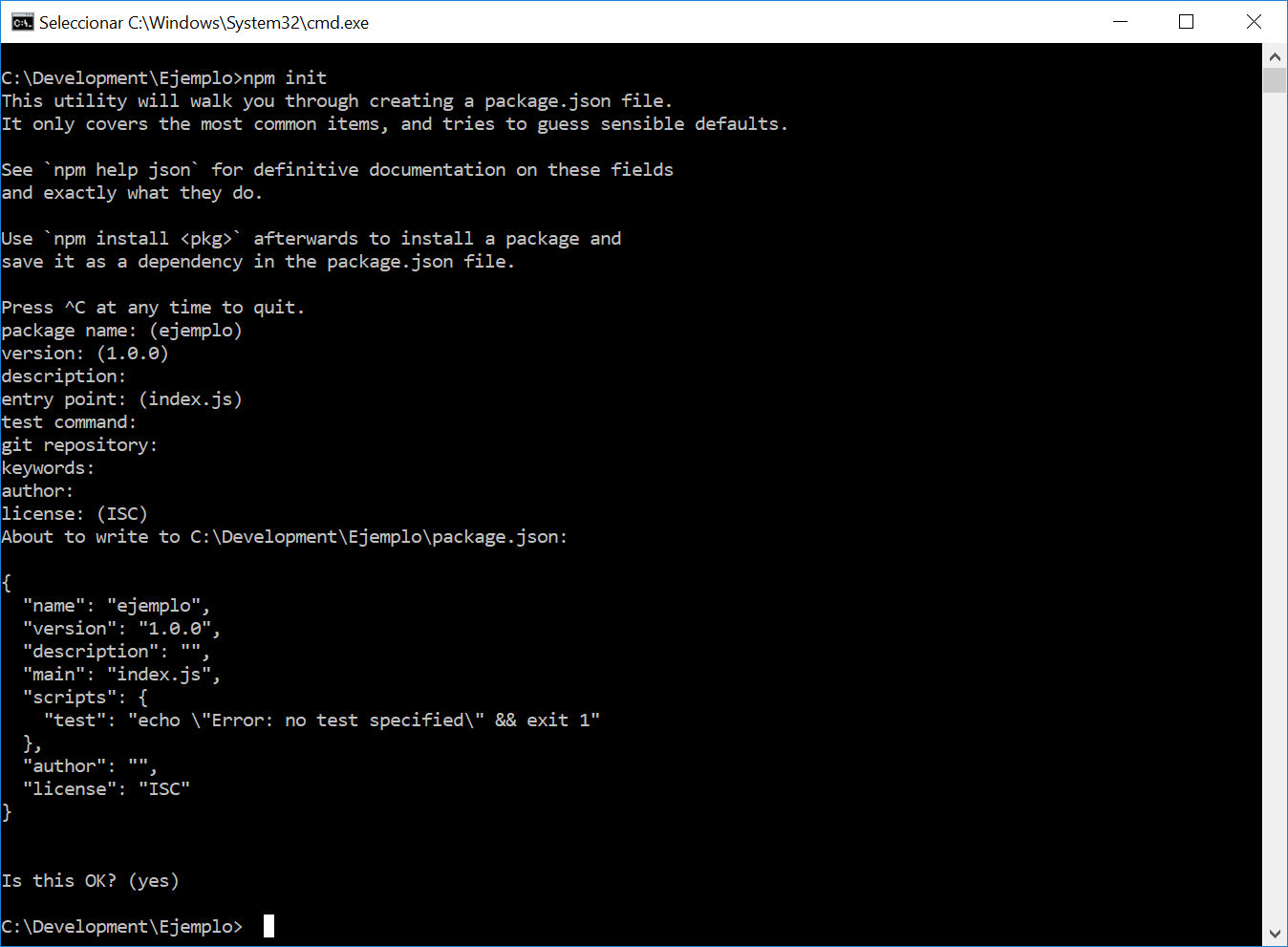
Introducción

Primero vamos a crear una carpeta que contendrá todo el proyecto en nuestro caso:

*C:/Development/Ejemplo*

Dentro de dicha carpeta abrimos la terminal y aplicamos el siguiente comando:

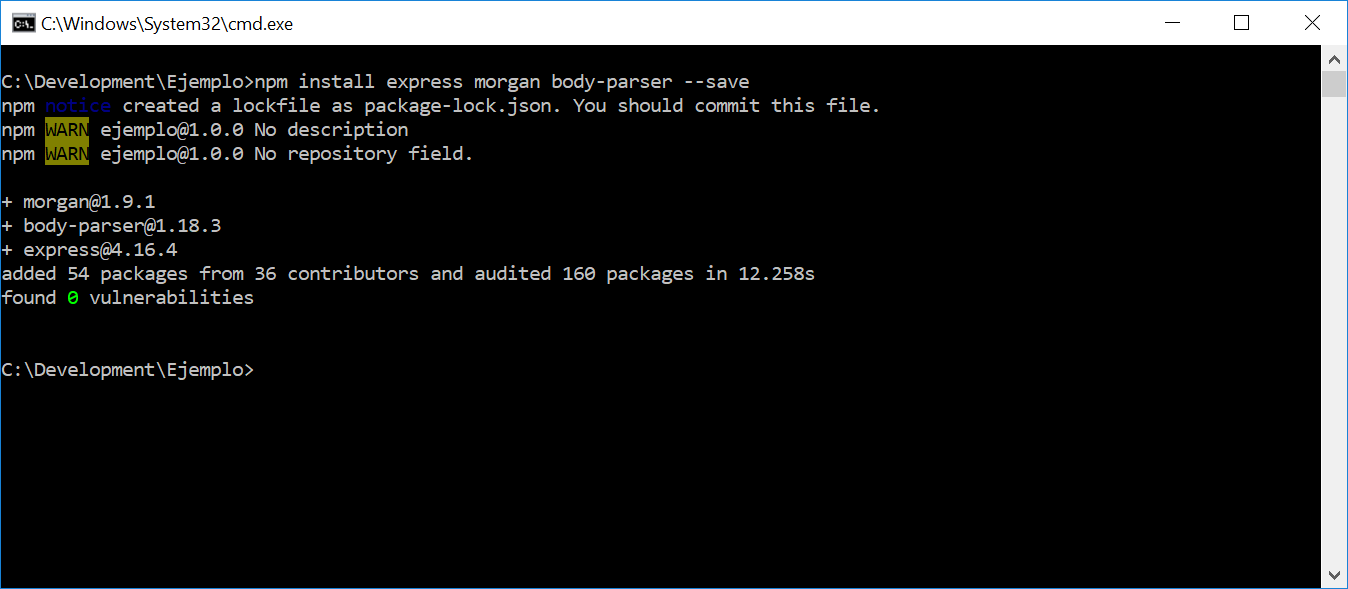
*npm init*



Una vez realizado esto se nos creará en nuestra carpeta “Ejemplo” el archivo *packaje.json*

El siguiente paso será instalar las dependencias con el siguiente comando:

*npm install express morgan body-parser –save*



Ahora creamos el archivo index.js o según el nombre que hayamos indicado en el primer paso en la opción de *entry point*, y lo colocamos en la raíz del proyecto.

Dentro de dicho archivo colocamos las siguientes instrucciones:

Constantes:

const express = require('express');

const app = express();

const logger = require('morgan');

const http = require('http');

const path = require('path');

const PORT = process.env.PORT || 8080;

const bodyParser = require('body-parser');

const baseAPI = '/api/v1';

Configuraciones:

app.use(bodyParser.json());

app.use(logger('dev'));

app.use(bodyParser.urlencoded({

extended: true

}));

Método GET:

app.get('/', function (req, res) {

res.send('Hello World!');

});

Instanciar el servidor y ponerlo a la escucha por el puerto 8080 (por defecto):

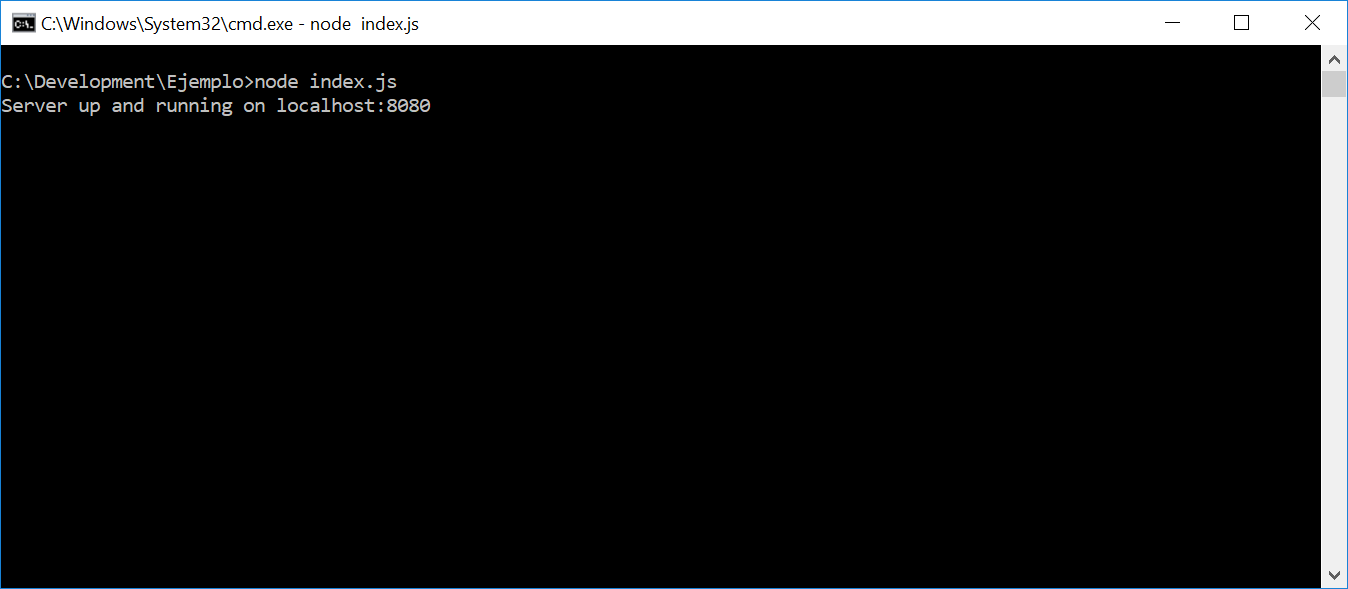
const server = http.createServer(app);

server.listen(PORT, function () {

console.log('Server up and running on localhost:' + PORT);

});

Para inicar la API rest nos colocamos en el directorio raíz del proyecto abrimos la consola y ejecutamos el siguiente comando:



Si todo se ha realizado correctamente el servidor quedaría lanzado y ahora crearemos la API de móviles para usarla como ejemplo para futuras APIs.

Para evitar sobrecargar el index.js vamos a optar por mostrar directamente como realizarlo usando *routes* que nos permite tener organizados los distintos recursos en nuestra API.

El primer paso será crear la carpeta routes en la raíz del proyecto:

*C:/Development/Ejemplo/routes*

Dentro de dicha carpeta creamos el archivo javascript según el recurso que queramos, en este caso, *mobiles.js*

*C:/Development/Ejemplo/routes/mobiles.js*

Dentro de dicho archivo añadimos las siguientes instrucciones:

'use strict';

const express = require('express');

const router = express.Router();

Ahora mismo no estamos usando persistencia así que trabajaremos sobre el siguiente JSON de mobiles, a continuación encontraríamos definidos los métodos GET, POST, DELETE, GET de un móvil que coincida con el criterio y PUT.

- GET devuelve todos los móviles del JSON.

- POST añade un móvil al JSON.

- DELETE borra todos los móviles del JSON.

- PUT modificar un móvil en concreto según la marca que indiquemos.

let mobiles = [

{

"marca":"Xiaomi",

"modelo":"Xiaomi Mi A2 4GB+64GB",

"precio":234

},

{

"marca": "Samsung",

"modelo": "Samsung s9",

"precio": 602

}

];

router.get('/', function (req, res) {

res.send(mobiles);

});

router.post('/', function (req, res) {

mobiles.push(req.body);

res.status(201).send("Mobiles created");

});

router.delete('/', function (req, res) {

mobiles = [];

res.status(200).send("Mobiles deleted");

});

router.get('/:marca', function (req, res) {

let marca = req.params.marca;

res.send(mobiles.filter(m => m.marca === marca));

});

router.put('/:marca', function (req, res) {

let marca = req.params.marca;

mobiles = mobiles.filter(m => m.marca !== marca);

mobiles.push(req.body);

res.status(200).send("Mobiles updated!");

});

Para exportar el router:

module.exports = router;

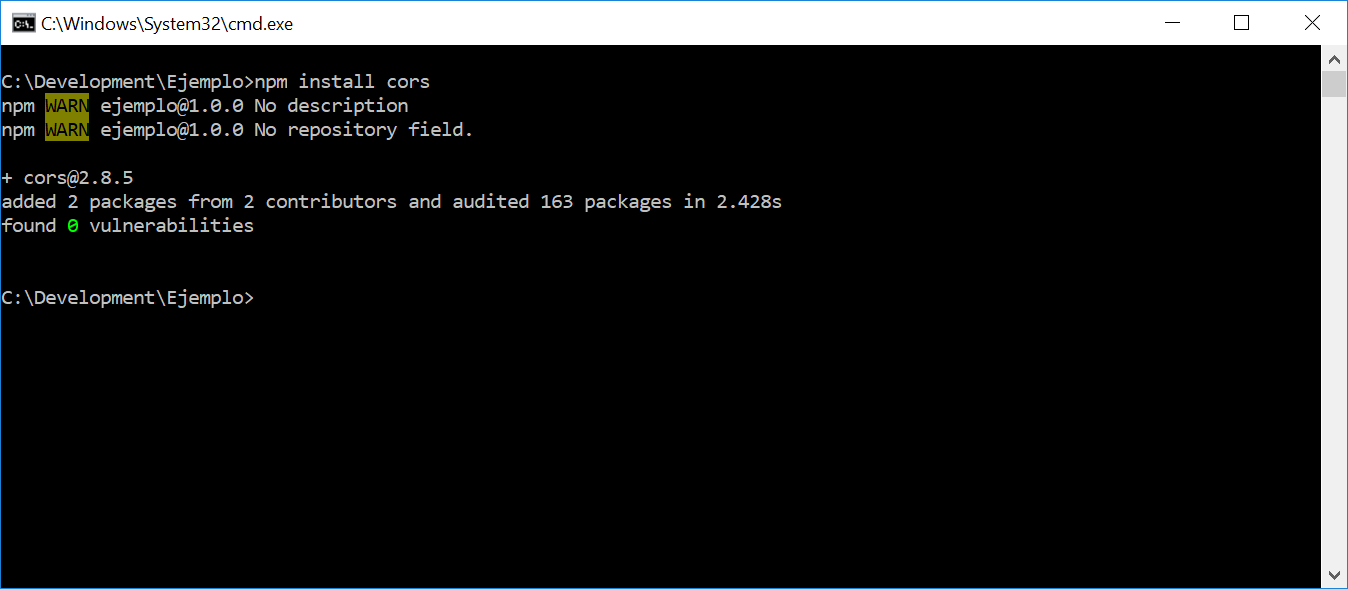
Añadimos al index.js las siguientes líneas para importar el router que acabamos de crear:

const mobiles = require('./routes/mobiles');

app.use('/mobiles', mobiles);

Instalamos después de esto las dependencias de CORS para que no deniegue las solicitudes que provengan de otro servidor:

*npm install cors –save*



Y añadimos las 2 siguientes líneas a nuestro index.js:

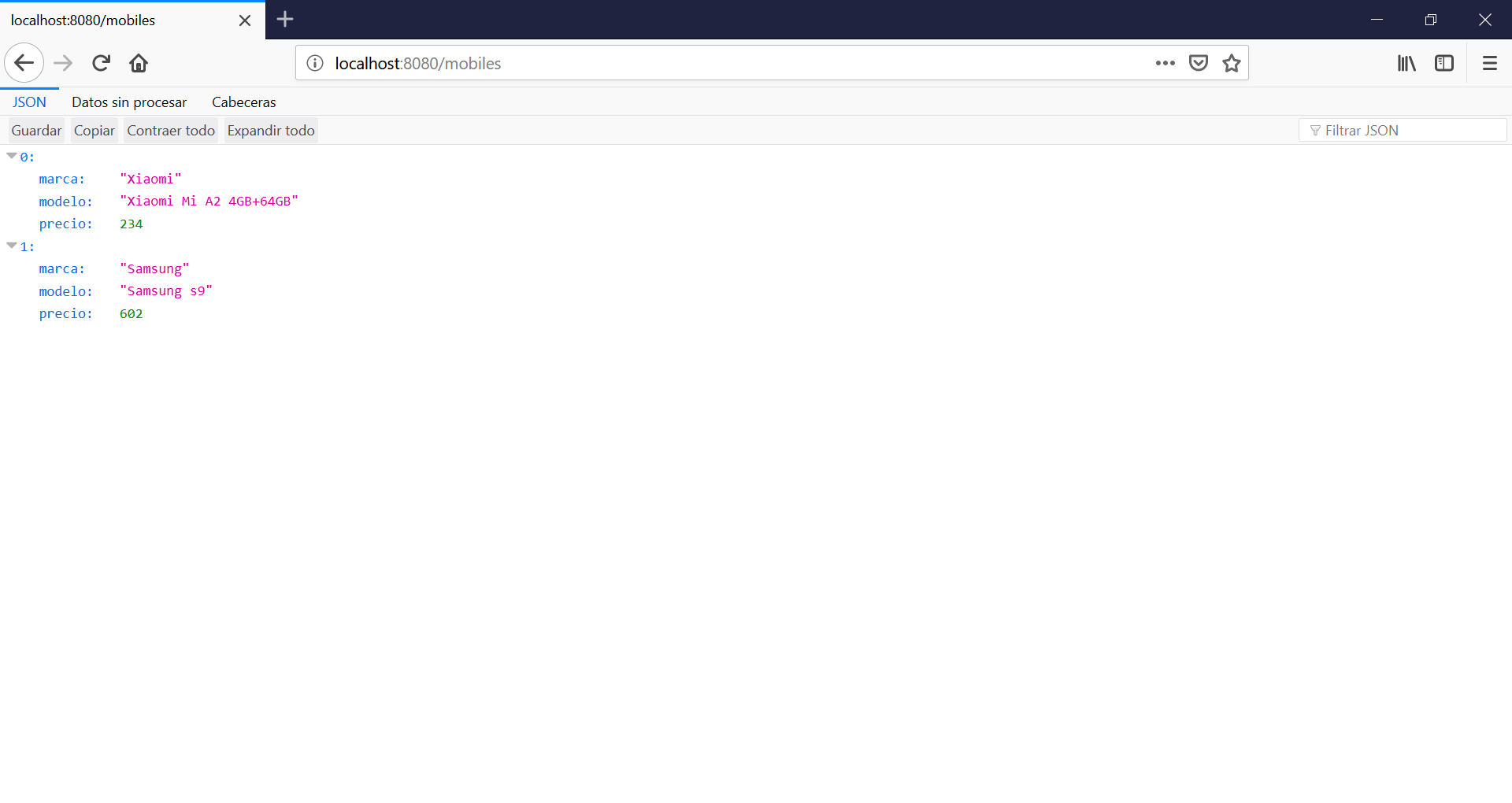
const cors = require('cors');

app.use(cors());

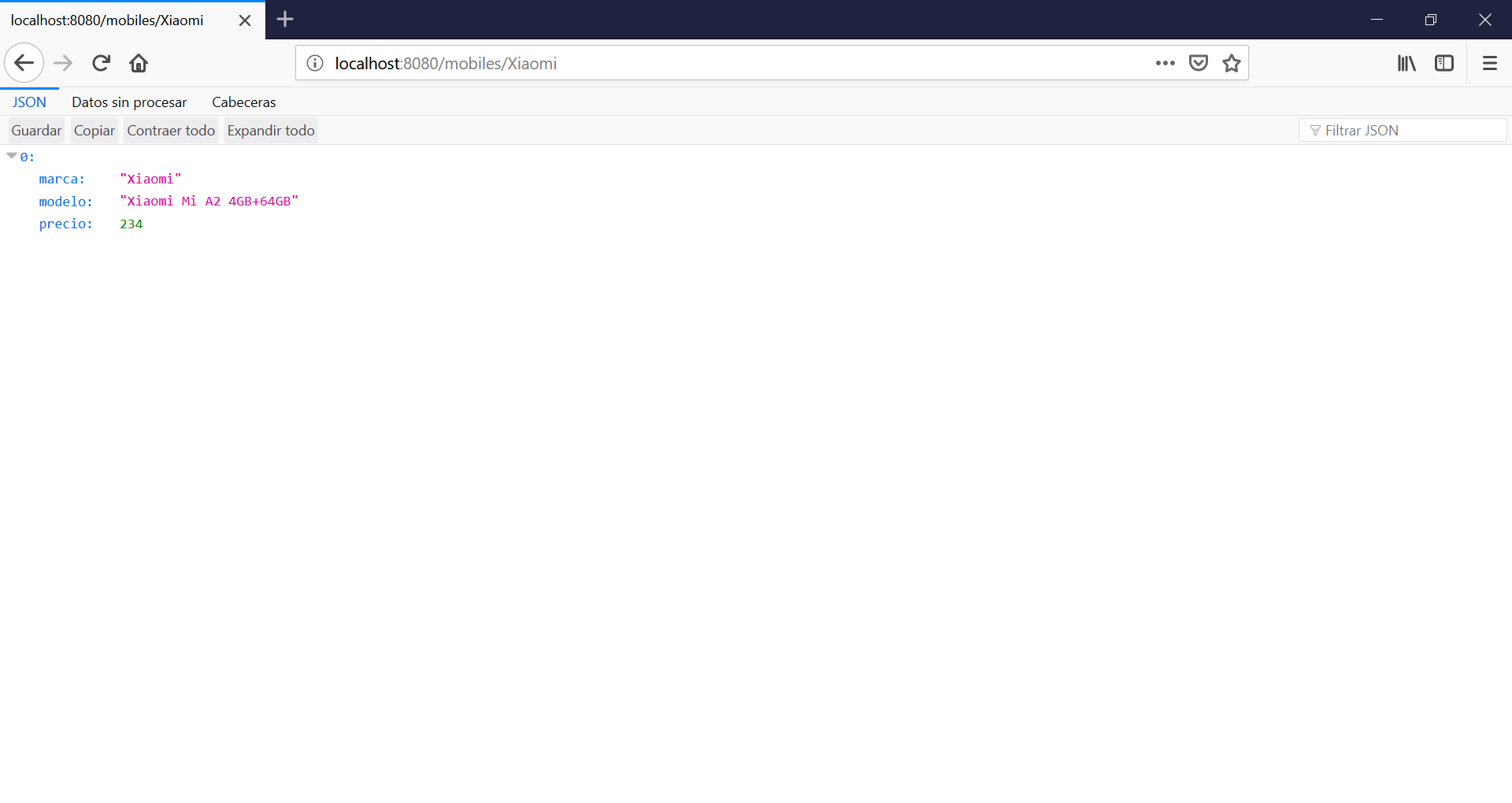
Una vez realizado todo esto empezaremos a probar las distintas peticiones a nuestro servidor:

* GET

Vamos a nuestro navegador y colocamos la siguiente URL: [*http://localhost:8080/mobiles*](http://localhost:8080/mobiles)

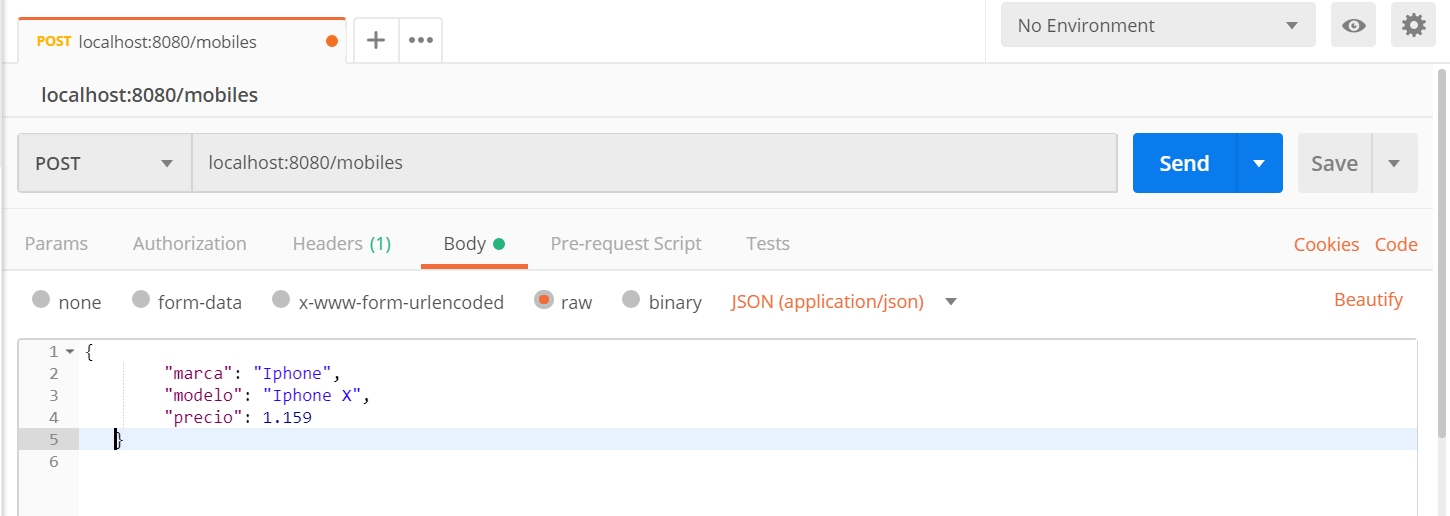


* GET con una marca en concreto:

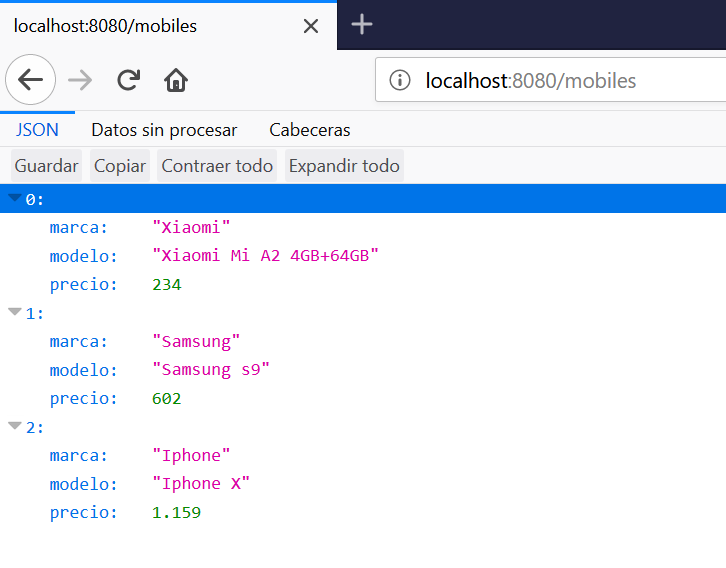


Para las demás operaciones utilizaremos la herramienta Postman:

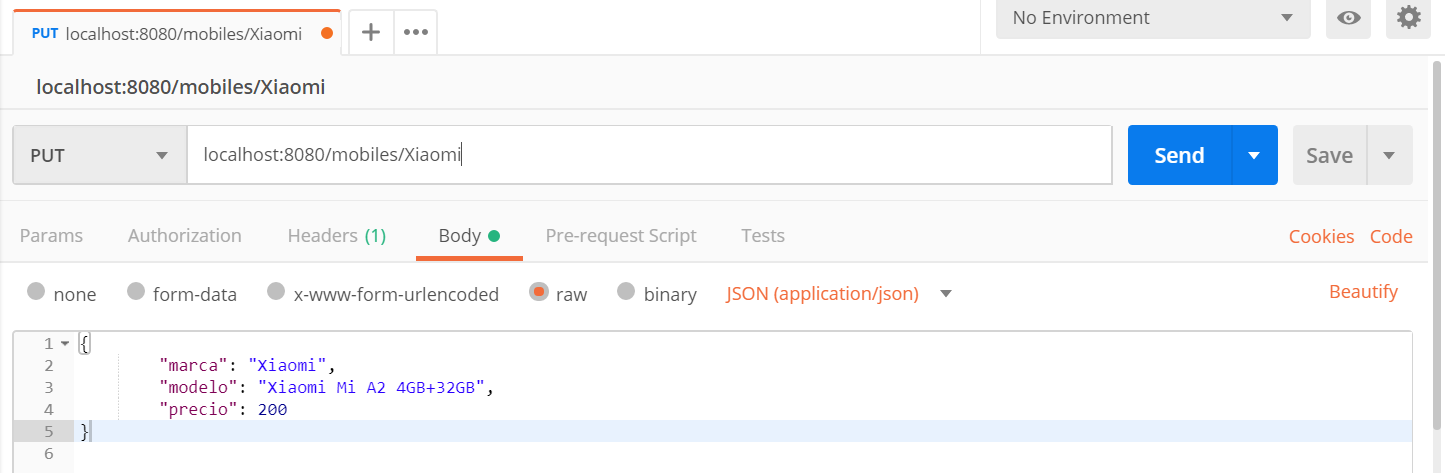
* POST

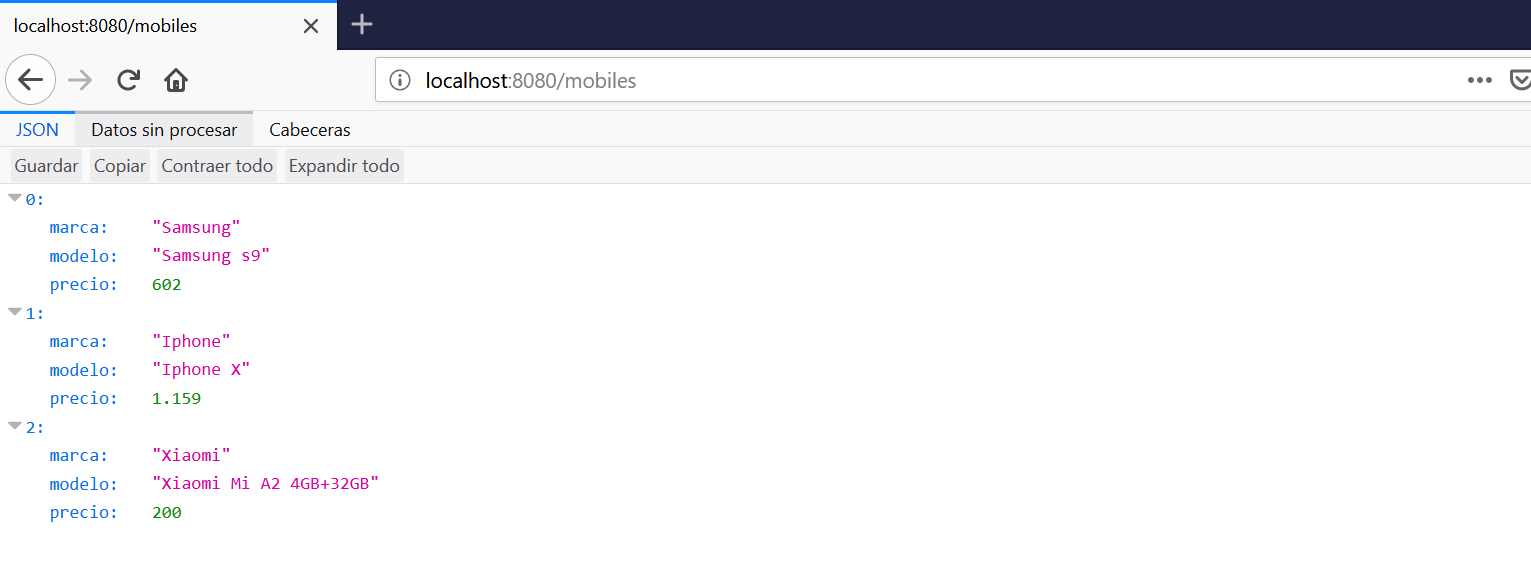


Y quedaría añadido:

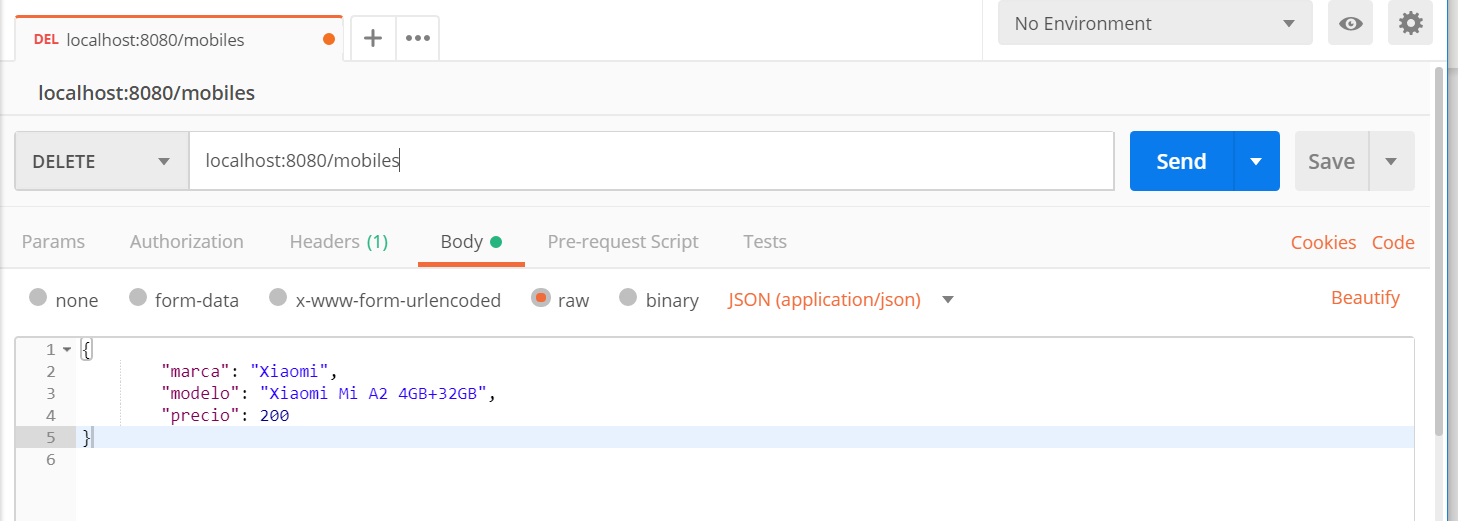


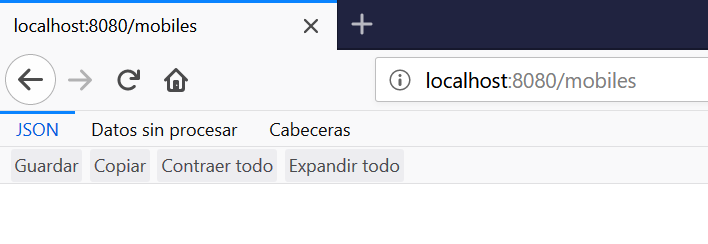
* PUT





* DELETE





Ahora falta que nuestra api tenga persistencia en sus datos. Para ello usaremos una base de datos, que crearemos usando MongoDB.

Una vez tengamos dicha base de datos, crearemos el script mobiles-service.js, que quedará situado en la siguiente ruta:

*C:/Development/Ejemplo/routes/mobiles-service.js*

Para inicializar la conexión con la base de datos que hemos creado, incluiremos en *mobiles-service.js* las siguientes instrucciones, rellenando con nuestro usuario, contraseña y el nombre correspondiente a la base de datos MongoDB:

'use strict';

const MongoClient = require('mongodb').MongoClient;

let db;

let ObjectId = require('mongodb').ObjectID;

const Mobiles = function () {

};

Mobiles.prototype.connectDb = function (callback) {

MongoClient.connect("mongodb://**USUARIO:CONTRASEÑA**@ds039351.mlab.com:39351/**NOMBREBD**",

{useNewUrlParser: true},

function (err, database) {

if (err) {

callback(err);

}

db = database.db('**NOMBREBD**').collection('mobiles');

callback(err, database);

});

};

Ahora añadimos los métodos necesarios para trabajar con nuestra base de datos:

Mobiles.prototype.add = function (mobile, callback) {

return db.insert(mobile, callback);

};

Mobiles.prototype.get = function (\_id, callback) {

return db.find({\_id:

ObjectId(\_id)}).toArray(callback);

};

Mobiles.prototype.getAll = function (callback) {

return db.find({}).toArray(callback);

};

Mobiles.prototype.update = function (\_id, updatedMobile, callback) {

delete updatedMobile.\_id;

return db.updateOne({\_id: ObjectId(\_id)}, {$set:

updatedMobile}, callback);};

Mobiles.prototype.remove = function (\_id, callback) {

return db.deleteOne({\_id: ObjectId(\_id)}, callback);

};

Mobiles.prototype.removeAll = function (callback) {

return db.deleteMany({}, callback);

};

module.exports = new Mobiles();

Ahora debemos reconfigurar *mobiles.js* con el objetivo de hacer que las operaciones se efectúen sobre la base de datos. En primer lugar, eliminaremos el array mobiles:

let mobiles = […]

Una vez eliminado *mobiles*, añadiremos la siguiente instrucción justo a continuación de la inicialización del router:

const moviesService = require('./movies-service');

El siguiente paso es la actualización de los métodos de *movies.js*, que deben quedar tal y como si indica a continuación:

ANTES:

router.get('/', function (req, res) {

res.send(mobiles);

});

DESPUÉS:

router.get('/', function (req, res) {

mobilesService.getAll((err, mobiles) => {

if (err) {

res.status(500).send({

msg: err

});

} else if (mobiles === null){

res.status(500).send({

msg: "mobiles null"

});

} else if (mobiles !== null) {

res.status(200).send(mobiles);

}

}

);

});

ANTES:

router.post('/', function (req, res) {

mobiles.push(req.body);

res.status(201).send("Mobiles created"); });

DESPUÉS:

router.post('/', function (req, res) {

let mobile = req.body;

mobilesService.add(mobile, (err, mobile) => {

if (err) {

res.status(500).send({

msg: err

});

} else if (mobile !== null) {

res.send({

msg: 'Mobile created!'

});

}

}

);

});

ANTES:

router.delete('/', function (req, res) {

mobiles = [];

res.status(200).send("Mobiles deleted");});

DESPUÉS:

router.delete('/', function (req, res) {

mobilesService.removeAll((err) => {

if (err) {

res.status(500).send({

msg: err

});

} else {

res.status(200).send({

msg: 'Mobiles deleted!'

});

}

});

});

ANTES:

router.get('/:marca', function (req, res) {

let marca = req.params.marca;

res.send(mobiles.filter(m => m.marca === marca));});

DESPUÉS:

router.get('/:\_id', function (req, res) {

let \_id = req.params.\_id;

mobilesService.get(\_id, (err, mobile) => {

if (err) {

res.status(500).send({

msg: err

});

} else if (mobile === null){

res.status(500).send({

msg: "mobiles null"

});

} else if (mobile !== null) {

res.status(200).send(mobile);

}

}

);

});

ANTES:

router.put('/:marca', function (req, res) {

let marca = req.params.marca;

mobiles = mobiles.filter(m => m.marca !== marca);

mobiles.push(req.body);

res.status(200).send("Mobiles updated!");

}

);

DESPUÉS:

router.put('/:\_id', function (req, res) {

const \_id = req.params.\_id;

const updatedMobile = req.body;

mobilesService.update(\_id, updatedMobile, (err,

numUpdates) => {

if (err || numUpdates === 0) {

res.status(500).send({

msg: err

});

} else {

res.status(200).send({

msg: 'Mobile updated!'

});

}

});

});

ANTES:

router.delete('/:modelo', function (req, res) {

let modelo = req.params.modelo;

mobiles = mobiles.filter(m => m.modelo !== modelo);

res.status(200).send("Mobile deleted!");

});

DESPUÉS:

router.delete('/:\_id', function (req, res) {

let \_id = req.params.\_id;

mobilesService.remove(\_id, (err) => {

if (err) {

res.status(404).send({

msg: err

});

} else {

res.status(200).send({

msg: 'Mobile deleted!'

});

}

});

});

Una vez modificados todos los métodos, importamos el fichero *movies-services.js* en nuestro *index.js* con la siguiente instrucción:

const moviesService = require('./routes/movies-service');

Y la función

server.listen(PORT, function () {

console.log('Server up and running on localhost:' + PORT);

});

la sustituimos por

moviesService.connectDb(function (err) {

if (err) {

console.log("Could not connect with MongoDB - moviesService");

process.exit(1);

}

server.listen(PORT, function () {

console.log('Server up and running on localhost:' + PORT);

});

});

Con todos estos cambios ya tendremos configurada nuestra conexión y operaciones, ahora abrimos una terminal y ejecutamos el siguiente comando para instalar MongoDB:

*npm install mongodb –save*

Una vez termine la instalación, solo tendremos que abrir una terminal y escribir el siguiente comando ejecutar el servidor en el localhost:8080:

*node index.js*