Ejercicios Tema 8

Ejercicio 1

En este ejercicio vamos a crear un completo sistema CRUD (create, read, update, delete) que nos permita interactuar con alguno de los modelos que vamos a crear dentro de nuestra base de datos. Con la estructura que vamos a desarrollar conseguiremos tener un API de nuestra web para permitir el acceso desde otro tipo de dispositivo o página.

Nuestro proyecto almacenará datos de tareas por hacer (To-Do) y podremos interactuar sobre ellas.

Para acelerar nuestro trabajo, vamos a ayudarnos de las herramientas que ya hemos visto anteriormente, como puede ser **express** y **mongoose**.

- Lo primero que vamos a hacer es crear el proyecto donde desarrollaremos nuestro CRUD.

  express EjercicioToDo

Es importante que no se nos olvide instalar las dependencias del proyecto (npm install) y que lo arranquemos para ver que todo va correctamente (npm start).

- Nuestro siguiente paso es instalar el módulo de mongoose:

  npm install mongoose —save

Con el flag save agregamos la dependencia del módulo de mongoose dentro del fichero de dependencias de nuestro proyecto, lo que nos permitirá poder hacer deploy del proyecto en cualquier servidor y mantendremos las dependencias de los módulos necesarios.

- Procedemos a crear el modelo con el que vamos a trabajar:

Dentro del directorio **models**, creamos el script **ToDo.js**

```
var mongoose = require ('mongoose');
var Schema = mongoose.Schema;

mongoose.connect('mongodb://localhost/test');

var ToDoSchema = new Schema({
     titulo: 'String',
```



```
    completa: 'Number'
});

module.exports = mongoose.model('ToDo', ToDoSchema);
```

- En nuestro script **app.js** tenemos que detallar dónde se van a tratar las diferentes peticiones de nuestra Api

```
var todo = require('./controllers/ToDoControlle
…

app.use('/api', todo);
```

- El siguiente paso consistirá en crear el controlador mediante el cual vamos a manejar los diferentes accesos a nuestra API. Dentro del directorio **controllers** vamos a generar el script **ToDoController.js**.

Definimos los diferentes require para nuestro script

```
var router = require('express').Router();
var ToDo = require('../models/ToDo.js');
```

Ahora pasamos a definir las diferentes rutas con las que vamos a trabajar. La primera sería si accedemos a través del método GET sin pasar nada por parámetro. Lo que queremos conseguir es que se listen todas las tareas que tenemos en nuestra base de datos.

```
router.get('/', function(req, res){
    // Obtenemos todas la tareas
    ToDo.find(function(err, list){
        // Comprobamos si podemos trabajar con el formato json
        if(req.accepts('json')){
            //Si encontramos algún tipo de error, lo notificamos
            if(err){
                return res.json(500, {message: 'Error obteniendo las
tareas'});
            }
            // Si no hay error montamos un json a partir de la lista
de tareas
            return res.json(list);
        }else{
            res.send ('No acepta JSON');
        }
    });
});
```

Para poder insertar tareas, lo haremos mandando una petición PUT sobre la raíz de nuestro api, pasándole el parámetro del título como clave.


```
router.post('/', function(req, res){
     // Creamos la tarea a partir de los parámetros de la petición
     var tarea = new ToDo({
          // En req.body tenemos todos los parámetros almacenados
          // en nuestra petición POST
          'titulo': req.body['titulo'],
          'completa': 0
     });
     // Salvamos la tarea que acabamos de crear y comprobamos
     // los errores.
     tarea.save(function(err, t){
          if(req.accepts('json')){
               if(err){
                    return res.json(500, {message: "Error guardando la
tarea", error: err});
               }
               return res.json({message: "Salvado", _id: tarea._id});
          }else{
               res.send("No acepta JSON");
          }
     });
});
```

Si queremos obtener una tarea de todas las insertadas dentro de la base de datos, podemos crear una petición GET en la que pasamos el identificador de la ruta que queremos obtener.

```
router.get('/:id', function(req, res){
     // En el objeto params podemos obtener los elementos
     // que obtengamos en la url
     var id = req.params.id;
     // Hacemos la búsqueda en la base de datos y manejamos los errores
     ToDo.findOne({_id:id}, function(err, tarea){
          if (req.accepts('json')){
               if(err){
                    return res.json(500, {message: 'Error obteniendo la
tarea'});
               }
               if(!tarea){
                    return res.json({message: "No se ha podido obtener la
tarea"});
               }
               return res.json(tarea);
          }else{
               res.send("No acepta JSON");
          }
     });
});
```


Al final de nuestro controlador, tendrás que exportar el objeto router que hemos estado usando:

```
module.exports = router;
```

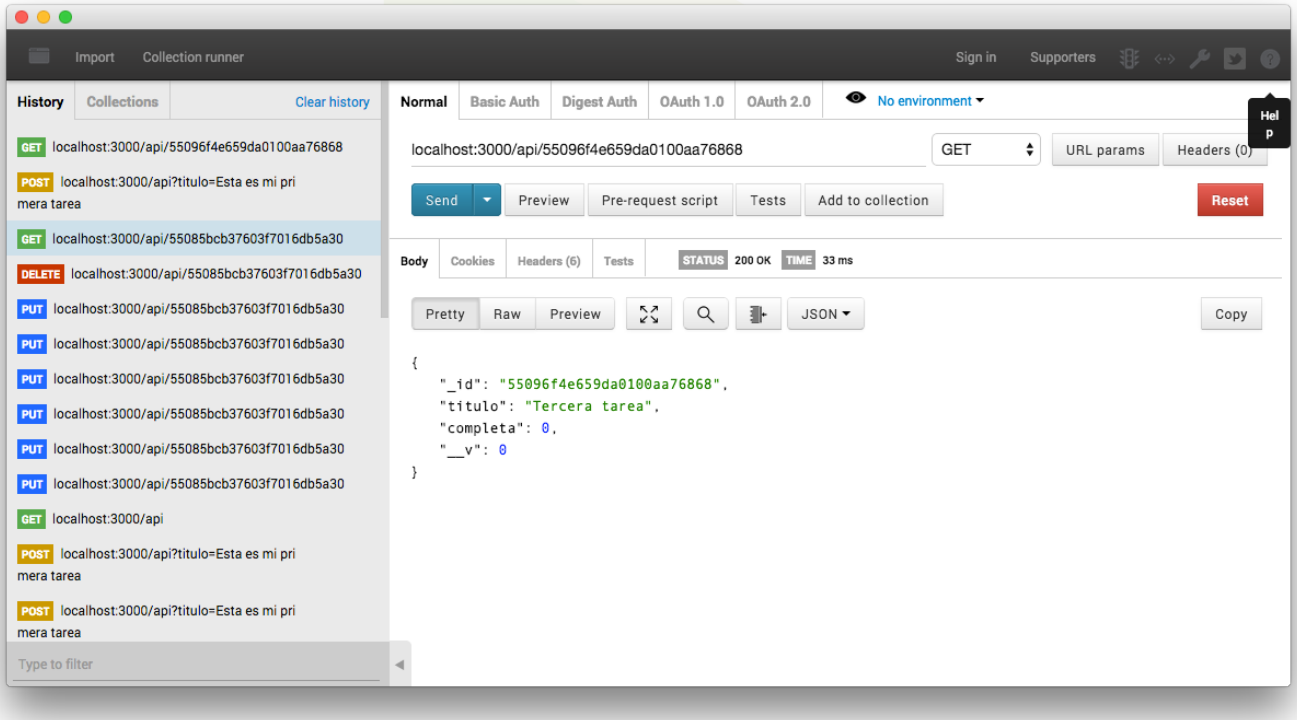
Ejercicio 2

Realiza las peticiones **PUT** y **DELETE** para actualizar una tarea en concreto (ten en cuenta que trabaja muy parecido a la petición POST) y borrar una tarea concreta respectivamente.

PRUEBAS

Para poder probar las diferentes peticiones, en vez de crear vistas con las cuales se pueda acceder a las diferentes rutas, podemos usar la aplicación de Chrome **Postman**, la cual nos permite lanzar las peticiones que necesitemos pasándole los parámetros necesarios.

http://www.getpostman.com/docs/launch



Por supuesto siempre tenemos la opción de utilizar cURL, pudiendo mejorar su aspecto por ejemplo con pjson (de tener python también instalado: pip install pjson):

```
curl --header "Accept: application/json" \
   http://localhost:3000/api/ | pjson
```

```
curl --header "Accept: application/json" \
 --header "Content-Type:application/json" \
 --request POST \
 --data '{"titulo":"Test Post","completa":0}' \
   http://localhost:3000/api/ | pjson


curl --header "Accept: application/json" \
   http://localhost:3000/api/5694e9766a5d4023042865aj | pjson


curl  --header "Content-Type:application/json" \
 --header "Accept: application/json" \
 --request PUT \
 --data '{"titulo":"Test PUT","completa":1}' \
   http://localhost:3000/api/5694e9766a5d4023042865aj | pjson

curl --header "Content-Type:application/json" \
 --header "Accept: application/json" \
 --request DELETE \
   http://localhost:3000/api/5694e9766a5d4023042865aj | pjson
```

