

API REST SIN PERSISTENCIA



Índice

¿Qué es REST?.....	Pag. 3
Arquitectura REST.....	Pag. 3 y 4
Diseño e implementación.....	Pag. 4 - 8
Problemas encontrados.....	Pag 8
Mejoras y conclusiones.....	Pag 8
Documentación.....	Pag 8
Recursos consultados.....	Pag 8

¿Qué es REST?

La **transferencia de estado representacional**, Es un tipo de arquitectura de desarrollo web que se apoya totalmente en el estándar HTTP. REST se compone de una lista de reglas que se deben cumplir en el diseño de la arquitectura de una API.

Arquitectura REST



Reglas de una arquitectura REST

Interfaz Uniforme

La interfaz se basa en recursos, por ejemplo el recurso Dishes (name, image, category, label, price, featured, description), o el recurso comments (rating, comment, author, date).

El servidor mandará los datos (vía json).

Separación de cliente y servidor

El cliente y servidor están separados, su unión es mediante importación de módulos.

En nuestro caso hemos diseñado los scripts **dishes.js** y **comments.js** con las funciones de selección, inserción, modificación y borrado, y luego hemos creado un script **server.js** donde hacemos llamada a los módulos requeridos y donde realizamos el enrutamiento para cada una de las funciones anteriores.

Cacheable

En la web los clientes pueden cachear las respuestas del servidor

Las respuestas se deben marcar de forma implícita o explícita como cacheables o no.

En futuras peticiones, el cliente sabrá si puede reutilizar o no los datos que ya ha obtenido.

Si ahorramos peticiones, mejoraremos la escalabilidad de la aplicación y el rendimiento en cliente (evitamos principalmente la latencia).

Peticiones sin estado

Http es un protocolo sin estado lo que proporciona un mayor rendimiento.

Diseño e implementación

Hemos implementado endpoints para añadir, eliminar, editar o consultar diferentes productos. Para hacer este servicio RESTful, hemos definido los siguientes métodos para las entidades Dishes y Comments:

POST: Añade nuevas entidades.

PUT: Modifica la entidad según el ID que hayamos proporcionado.

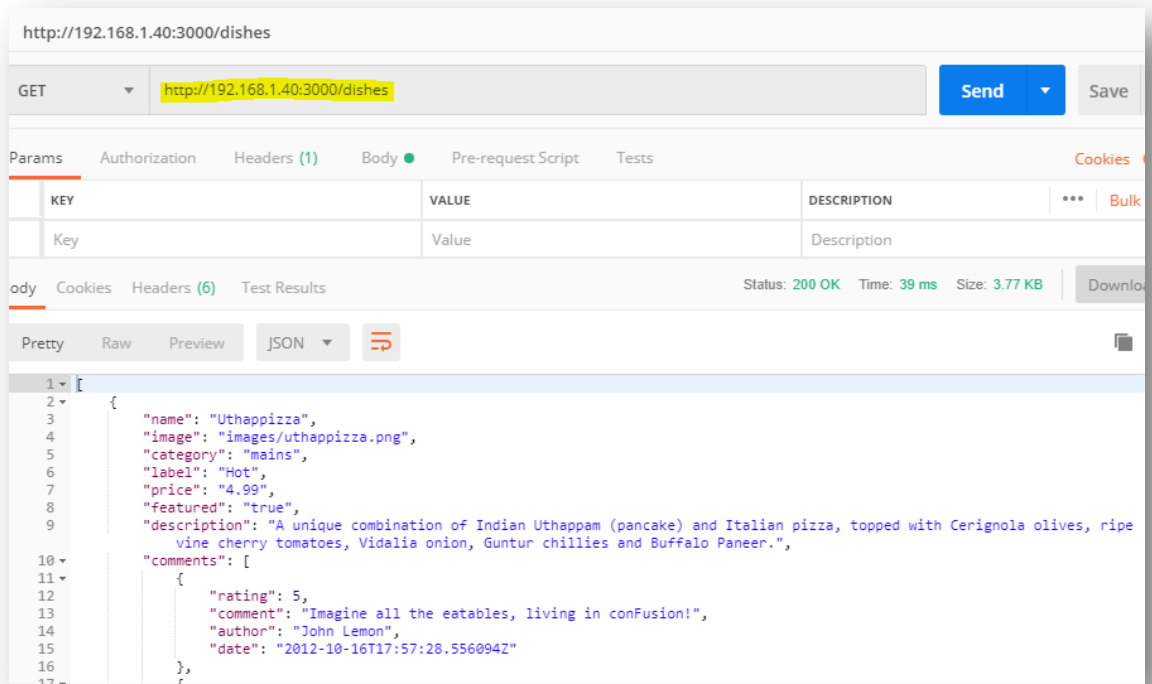
GET: Obtenemos el resultado de todas las entidades o de alguna en concreto si especificamos un id.

DELETE: Eliminamos la entidad que contenga el id indicado.

La aplicación no dispone de interfaz, por lo que las consultas se realizarán mediante postman proporcionando urls con los datos necesarios para realizar cualquiera de las funciones definidas anteriormente. Vamos a especificar algunos ejemplos:

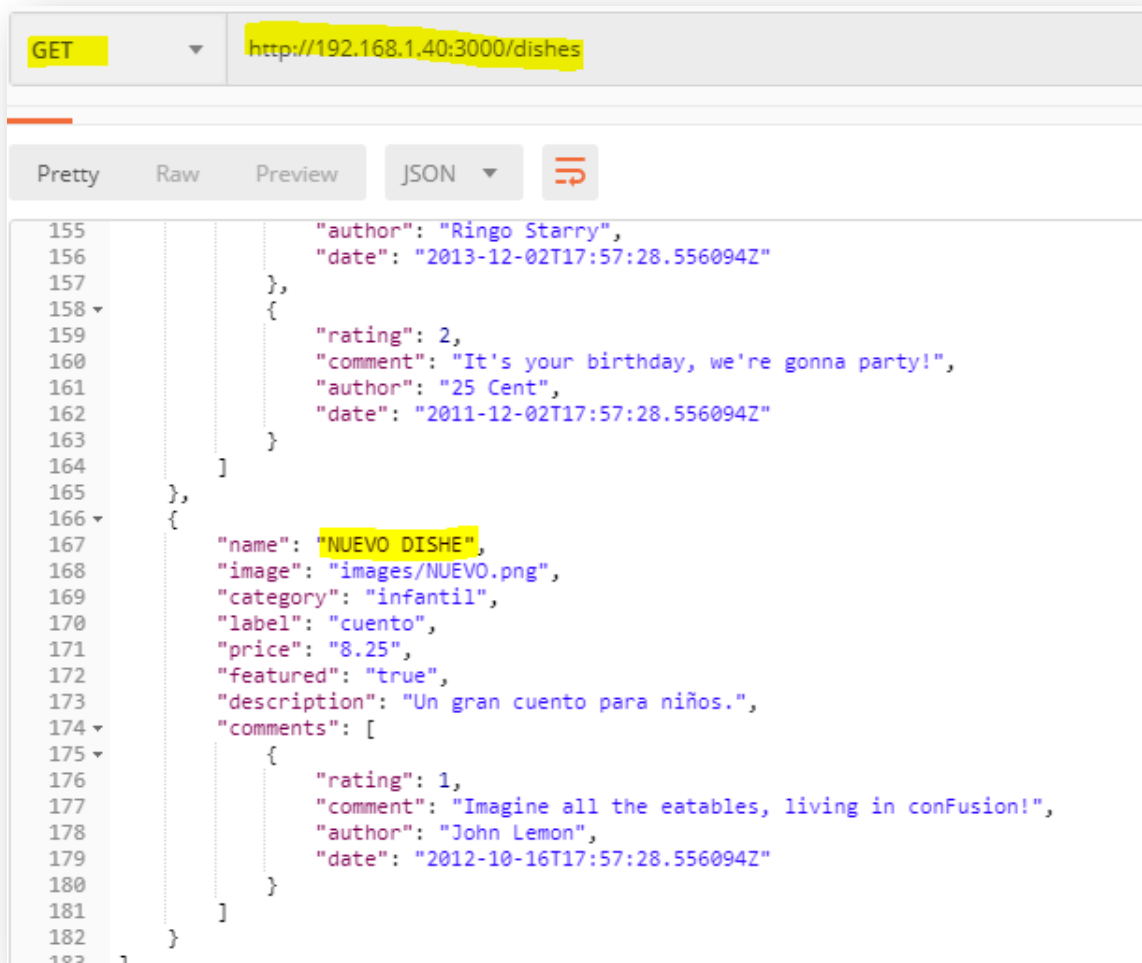
Para que funcione correctamente, se debe ejecutar script server.js que estará a la escucha en el puerto 3000.

GET. Obtener Dishes y Comments:



POST. Añadir Dishes con los siguientes valores:

```
{  "name": "NUEVO DISHE",
  "image": "images/NUEVO.png",
  "category": "infantil",
  "label": "cuento",
  "price": "8.25",
  "featured": "true",
  "description": "Un gran cuento para niños.",
  "comments": [
    {
      "rating": 1,
      "comment": "Imagine all the eatables, living in conFusion!",
      "author": "John Lemon",
      "date": "2012-10-16T17:57:28.556094Z"
    }
  ]
}
```



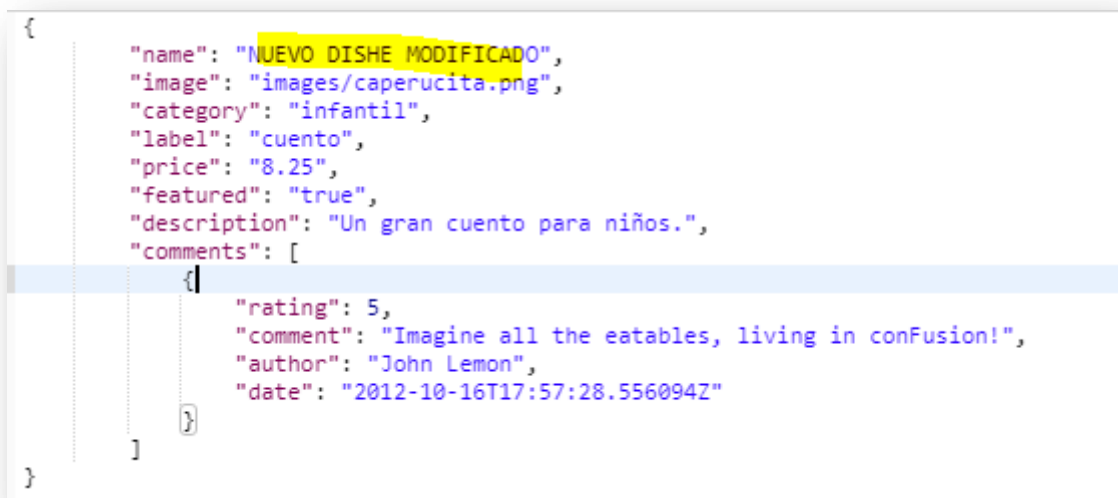
```
GET http://192.168.1.40:3000/dishes

Pretty Raw Preview JSON ↵

155     "author": "Ringo Starry",
156     "date": "2013-12-02T17:57:28.556094Z"
157   },
158   {
159     "rating": 2,
160     "comment": "It's your birthday, we're gonna party!",
161     "author": "25 Cent",
162     "date": "2011-12-02T17:57:28.556094Z"
163   }
164 ]
165 },
166 {
167   "name": "NUEVO DISHE",
168   "image": "images/NUEVO.png",
169   "category": "infantil",
170   "label": "cuento",
171   "price": "8.25",
172   "featured": "true",
173   "description": "Un gran cuento para niños.",
174   "comments": [
175     {
176       "rating": 1,
177       "comment": "Imagine all the eatables, living in confusion!",
178       "author": "John Lemon",
179       "date": "2012-10-16T17:57:28.556094Z"
180     }
181   ]
182 }
183 ]
```

PUT. Modificar Dishe:

Valores a modificar:



```
{
  "name": "NUEVO DISHE MODIFICADO",
  "image": "images/caperucita.png",
  "category": "infantil",
  "label": "cuento",
  "price": "8.25",
  "featured": "true",
  "description": "Un gran cuento para niños.",
  "comments": [
    {
      "rating": 5,
      "comment": "Imagine all the eatables, living in confusion!",
      "author": "John Lemon",
      "date": "2012-10-16T17:57:28.556094Z"
    }
  ]
}
```

```

GET http://192.168.1.40:3000/dishes

Pretty Raw Preview JSON

155     "author": "Ringo Starry",
156     "date": "2013-12-02T17:57:28.556094Z"
157   },
158   {
159     "rating": 2,
160     "comment": "It's your birthday, we're gonna party!",
161     "author": "25 Cent",
162     "date": "2011-12-02T17:57:28.556094Z"
163   }
164 ]
165 },
166 {
167   "name": "NUEVO DISHE MODIFICADO",
168   "image": "images/caperucita.png",
169   "category": "infantil",
170   "label": "cuento",
171   "price": "8.25",
172   "featured": "true",
173   "description": "Un gran cuento para niños.",
174   "comments": [
175     {
176       "rating": 5,
177       "comment": "Imagine all the eatables, living in conFusion!",
178       "author": "John Lemon",
179       "date": "2012-10-16T17:57:28.556094Z"
180     }
181   ]
182 }
183 ]
  
```

DELETE. Eliminar Dishe:

```

DELETE http://192.168.1.40:3000/dishes/4
  
```

```

GET http://192.168.1.40:3000/dishes/4

Params Authorization Headers (1) Body Pre-request Script
KEY VALUE
Key Value

Body Cookies Headers (7) Test Results
Pretty Raw Preview HTML

1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8">
5     <title>Error</title>
6   </head>
7   <body>
8     <pre>Cannot GET /dishes/4</pre>
9   </body>
10 </html>
  
```

El error es provocado porque no existe ese id ya que ha sido eliminado.

Todo estos ejemplos que hemos visto se pueden emplear para el elemento comments usando la url

<http://192.168.1.40:3000/dishes/:dishesid/comments/:commentsid>

Problemas encontrados

En la realización de este proyecto los errores encontrados no pasan de los errores de mala sintaxis.

Mejoras y conclusiones

Este proyecto puede obtener mejoras como la de usar un sistema de base de datos como puede ser MongoDB para almacenar los datos y que no se eliminen cada vez que se pare el servidor y vuelva a ejecutarse.

También sería bueno dotar a la aplicación de una interfaz más amigable donde pueda acceder cualquier tipo de usuario a realizar sus consultas sin tener que pararse a mirar cómo debe hacerlo, etc.

Documentación

El código fuente de este proyecto se encuentra subido en github. Debe acceder a la siguiente URL:

<https://github.com/kike094c/ProyectoDPL>

Debe descargarlo y colocarlo en un servidor con node js y express instalado. Ejecutar el archivo server.js y mediante consola o con la aplicación postman realizar las consultas GET, POST, PUT o DELETE.

Recursos consultados

<https://www.paradigmadigital.com/dev/definir-prototipar-api-rest/>

https://es.wikipedia.org/wiki/Transferencia_de_Estado_Representacional