

Documentación Api Coches Symfony

Hecho Por: José Carlos Pérez González

+Coches:

-Obtener todos los Coches:

URL → dominio/cars/get

Devuelve un array de coches

The screenshot shows a REST client interface. The query tab is active, displaying a GET request to `http://localhost:34831/cars/get`. The response tab shows a 200 OK status with a size of 1.03 KB and a time of 60 ms. The response body is a JSON array of four car objects.

```
1 {
2   {
3     "id": 6,
4     "model": "Golf R MK7",
5     "manufacturer": "Volkswagen",
6     "features": "310CV - 5-6 Marchas",
7     "price": "34000"
8   },
9   {
10    "id": 10,
11    "model": "Focus",
12    "manufacturer": "Ford",
13    "features": "125 caballos / 6 marchas",
14    "price": "22000"
15  },
16  {
17    "id": 11,
18    "model": "A3",
19    "manufacturer": "Audi",
20    "features": "180 caballos / 7 marchas",
21    "price": "35000"
22  },
23  {
24    "id": 12,
25    "model": "3 Series",
26    "manufacturer": "BMW",
27    "features": "200 caballos / 8 marchas",
28    "price": "40000"
29  },
30  {
31    "id": 13,
32    "model": "C-Class",
```

-Obtener un Coche mediante su Id:

URL → dominio/cars/get/{id}

Devuelve un array con un coche

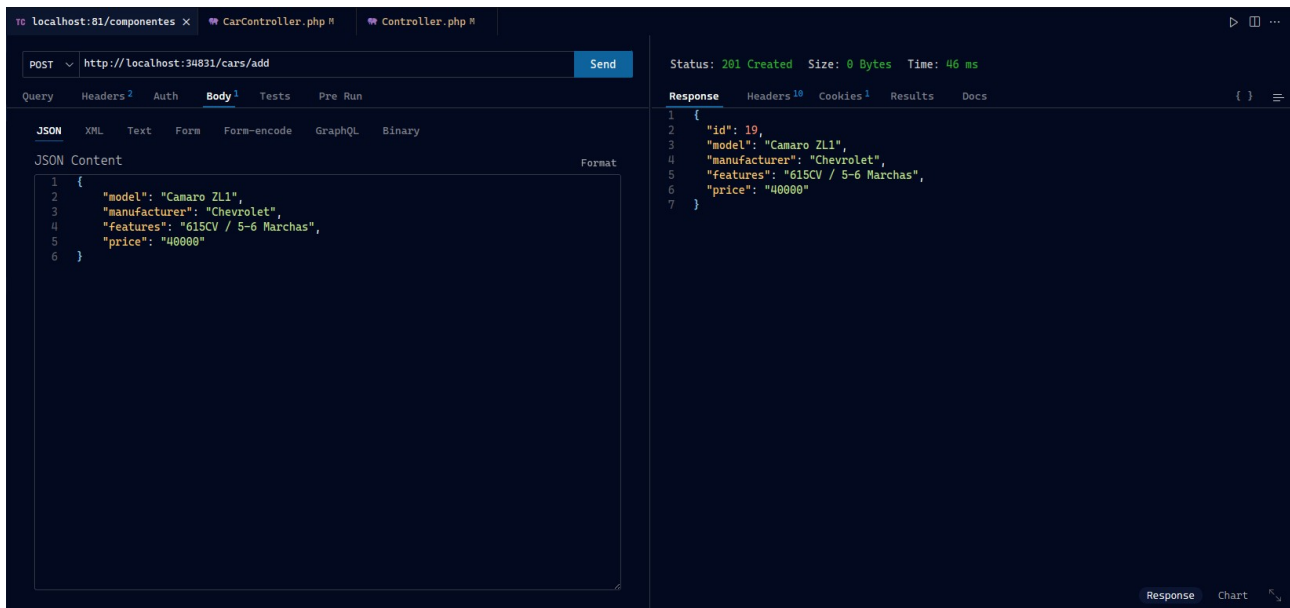
The screenshot shows a REST client interface. The query tab is active, displaying a GET request to `http://localhost:34831/cars/get/6`. The response tab shows a 200 OK status with a size of 108 Bytes and a time of 32 ms. The response body is a JSON array containing a single car object.

```
1 {
2   {
3     "id": 6,
4     "model": "Golf R MK7",
5     "manufacturer": "Volkswagen",
6     "features": "310CV - 5-6 Marchas",
7     "price": "34000"
8   }
9 }
```

-Añadir un Coche:

URL → dominio/cars/add

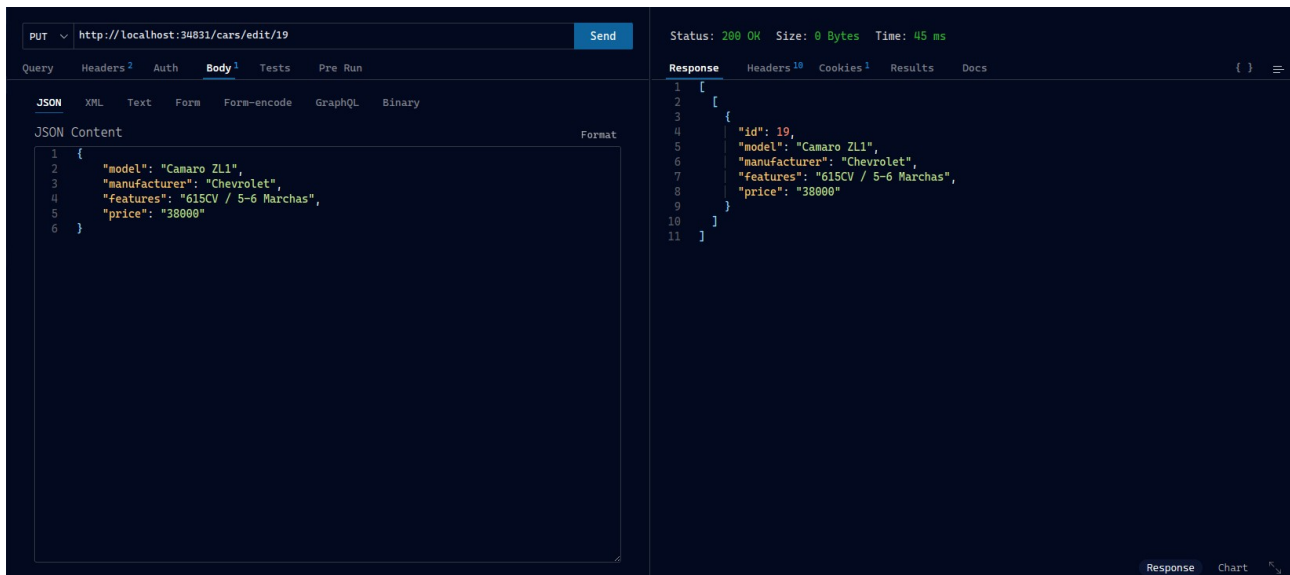
Devuelve el coche insertado



-Modificar todo un Coche mediante su Id:

URL → dominio/cars/edit/{id}

Devuelve un array con el coche ya modificado



-Modificar una propiedad de un Coche mediante su Id:

URL → dominio/cars/edit/{id}

Devuelve lo mismo que el get de antes

PUT <http://localhost:34831/cars/edit/19> Send

Status: 200 OK Size: 111 Bytes Time: 58 ms

Query Headers² Auth Body¹ Tests Pre Run

JSON XML Text Form Form-encode GraphQL Binary

JSON Content Format

```
1 {
2   "price": "38500"
3 }
```

Response Headers⁹ Cookies Results Docs

```
1 [
2   {
3     "id": 19,
4     "model": "Canaro ZL1",
5     "manufacturer": "Chevrolet",
6     "features": "615CV / 5-6 Marchas",
7     "price": "38500"
8   }
9 ]
10 ]
11 ]
```

Response Chart

-Eliminar un Coche mediante su Id:

URL → dominio/cars/delete/{id}

Devuelve el mensaje de confirmación de eliminación

DELETE <http://localhost:34831/cars/delete/15> Send

Status: 200 OK Size: 17 Bytes Time: 46 ms

Query Headers² Auth Body¹ Tests Pre Run

Query Parameters

parameter	value
<input type="checkbox"/>	

Response Headers⁹ Cookies Results Docs

```
1 "Coche Eliminado"
```

Response Chart

-Intentar eliminar un Coche mediante un Id inexistente:

URL → dominio/cars/delete/{id}

Devuelve el mensaje de aviso de que no se ha encontrado

The screenshot shows a REST client interface. The top bar indicates a GET request to `http://localhost:34831/cars/get/20`. The left pane shows the query parameters section, which is currently empty. The right pane shows the response details: Status: 200 OK, Size: 21 Bytes, Time: 41 ms. The response body contains the text `"Coche no encontrado"`.

+Usuarios:

-Obtener todos los Usuarios:

URL → dominio/user/get

Devuelve un array de usuarios

The screenshot shows a REST client interface. The top bar indicates a GET request to `http://localhost:34831/user/get`. The left pane shows the query parameters section, which is currently empty. The right pane shows the response details: Status: 200 OK, Size: 390 Bytes, Time: 44 ms. The response body contains a JSON array of five user objects, each with an email and a list of roles (ROLE_ADMIN and ROLE_USER).

```
1 [
2   {
3     "email": "joseca@gmail.com",
4     "roles": [
5       "ROLE_ADMIN",
6       "ROLE_USER"
7     ]
8   },
9   {
10    "email": "dani@gmail.com",
11    "roles": [
12      "ROLE_ADMIN",
13      "ROLE_USER"
14    ]
15  },
16  {
17    "email": "peri3@gmail.com",
18    "roles": [
19      "ROLE_USER"
20    ]
21  },
22  {
23    "email": "melli@gmail.com",
24    "roles": [
25      "ROLE_USER"
26    ]
27  },
28  {
29    "email": "admin@gmail.com",
30    "roles": [
31      "ROLE_ADMIN",
32      "ROLE_USER"
33    ]
34  }
35 ]
```

-Obtener un Usuario mediante su Id:

URL → dominio/user/get/{id}

Devuelve un array con el usuario

The screenshot shows a REST client interface. The top bar displays the method **GET** and the URL `http://localhost:34831/user/get/1`. The **Send** button is visible. Below the URL bar, the **Query** tab is active, showing a table for query parameters with columns **parameter** and **value**. The **Response** tab is also active, showing the status **200 OK**, size **65 Bytes**, and time **34 ms**. The response body is a JSON array containing one object with the following structure:

```
1 [
2   {
3     "email": "joseca@gmail.com",
4     "roles": [
5       "ROLE_ADMIN",
6       "ROLE_USER"
7     ]
8   }
9 ]
```

-Registrar un Usuario:

URL → dominio/user/register

Devuelve la frase de confirmación de registro

The screenshot shows a REST client interface. The top bar displays the method **POST** and the URL `http://localhost:34831/user/register`. The **Send** button is visible. Below the URL bar, the **Body** tab is active, showing the **JSON** content type. The request body is a JSON object with the following structure:

```
1 {
2   "email": "pruebaApi@gmail.com",
3   "password": "api"
4 }
```

The **Response** tab is also active, showing the status **201 Created**, size **0 Bytes**, and time **36 ms**. The response body is a plain text string:

```
1 "Usuario Creado Correctamente"
```

Usuario Registrado

The screenshot shows a REST client interface with a GET request to `http://localhost:34831/user/get`. The response is a JSON array of four user objects. The status is 200 OK, size is 444 Bytes, and time is 43 ms.

```
GET http://localhost:34831/user/get
```

Query Parameters

parameter	value
-----------	-------

Response

```
26 },
27 },
28 {
29   "email": "admin@gmail.com",
30   "roles": [
31     "ROLE_ADMIN",
32     "ROLE_USER"
33   ]
34 },
35 {
36   "email": "cafe@outlook.com",
37   "roles": [
38     "ROLE_USER"
39   ]
40 },
41 {
42   "email": "hola@gmail.com",
43   "roles": [
44     "ROLE_USER"
45   ]
46 },
47 {
48   "email": "pruebaApi@gmail.com",
49   "roles": [
50     "ROLE_USER"
51   ]
52 }
53 ]
```

-Comprobar los datos para el inicio de sesión de un Usuario:

Se le pasa mediante JSON los campos email y password para que compruebe si ha introducido correctamente los datos

URL → dominio/checklogin

Devuelve un JSON con una clave “value” y su valor es si ha introducido o no los datos correctos, y el otro son los roles del usuario con el correo indicado

The screenshot shows a REST client interface with a POST request to `http://localhost:34831/user/checklogin`. The request body is a JSON object containing email and password. The response is a JSON object with a value and a role. The status is 200 OK, size is 37 Bytes, and time is 35 ms.

```
POST http://localhost:34831/user/checklogin
```

JSON Content

```
1 {
2   "email": "pruebaApi@gmail.com",
3   "password": "api"
4 }
```

Response

```
1 {
2   "value": "true",
3   "role": [
4     "ROLE_USER"
5   ]
6 }
```