

DOCUMENTACIÓN API RESTFUL

ARTASHES ASATRYAN

Forma de acceso

Todos los métodos indicados en el documento se forman desde la URL base "<http://localhost:8080/api/v1>"

Inicio de sesión

Recurso	Descripción
POST auth/signup	Crea una cuenta con el email y contraseña que se pase en el body.
POST auth/signin	Se inicia sesión con el email y contraseña que se pase en el body.

Coches

Recurso	Descripción
GET coche	Devuelve todos los coches que hay en la base de datos.
GET coche/{id}	Devuelve el coche que tenga el id pasado.
POST coche	Crea el coche pasado por body.
PUT coche/{id}	Actualiza el coche con el id buscado con los datos pasados por el body.
DELETE coche/{id}	Borra el coche con el id buscado.

Motores

Recurso	Descripción
GET motor	Devuelve todos los motores que hay en la base de datos.
GET motor/{id}	Devuelve el motor que tenga el id

	pasado.
POST motor	Crea el motor pasado por body.
PUT motor/{id}	Actualiza el motor con el id buscado con los datos pasados por el body.
DELETE motor/{id}	Borra el motor con el id buscado.

Configuración

Para nuestro proyecto que se va a ejecutar en el puerto 8080 local, vamos a utilizar el entorno "Concesionario":

Concesionario Fork 0

Filter variables

	Variable	Type	Initial value	Current value
<input checked="" type="checkbox"/>	URL_BASE	default	http://localhost:8080/api/v1	http://localhost:8080/api/v1
	Add new varia...			

Vamos a tener los siguientes tests:

- ▼ CONCESIONARIO
- POST signup
 - POST signin
 - GET obtener coche por id
 - GET obtener coches
 - GET obtener motores
 - GET obtener motor por id
 - PUT Actualizar coche
 - PUT Actualizar motor
 - DEL Borrar motor
 - DEL Borrar coche
 - POST crear coche
 - POST crear motor

Crear cuenta

Primero, necesitaremos crear una cuenta. Para ello nos dirigimos al enlace de signup, pasamos el json del email y contraseña por el body y hacemos un POST:

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `{{URL_BASE}}/auth/signup`
- Body Type:** JSON
- Request Body:**

```
1 {
2   "email": "ejemplo1@ejemplo.com",
3   "password": "ejemplo"
4 }
```
- Response:** 200 OK, 78 ms, 780 B
- Response Body (Pretty):**

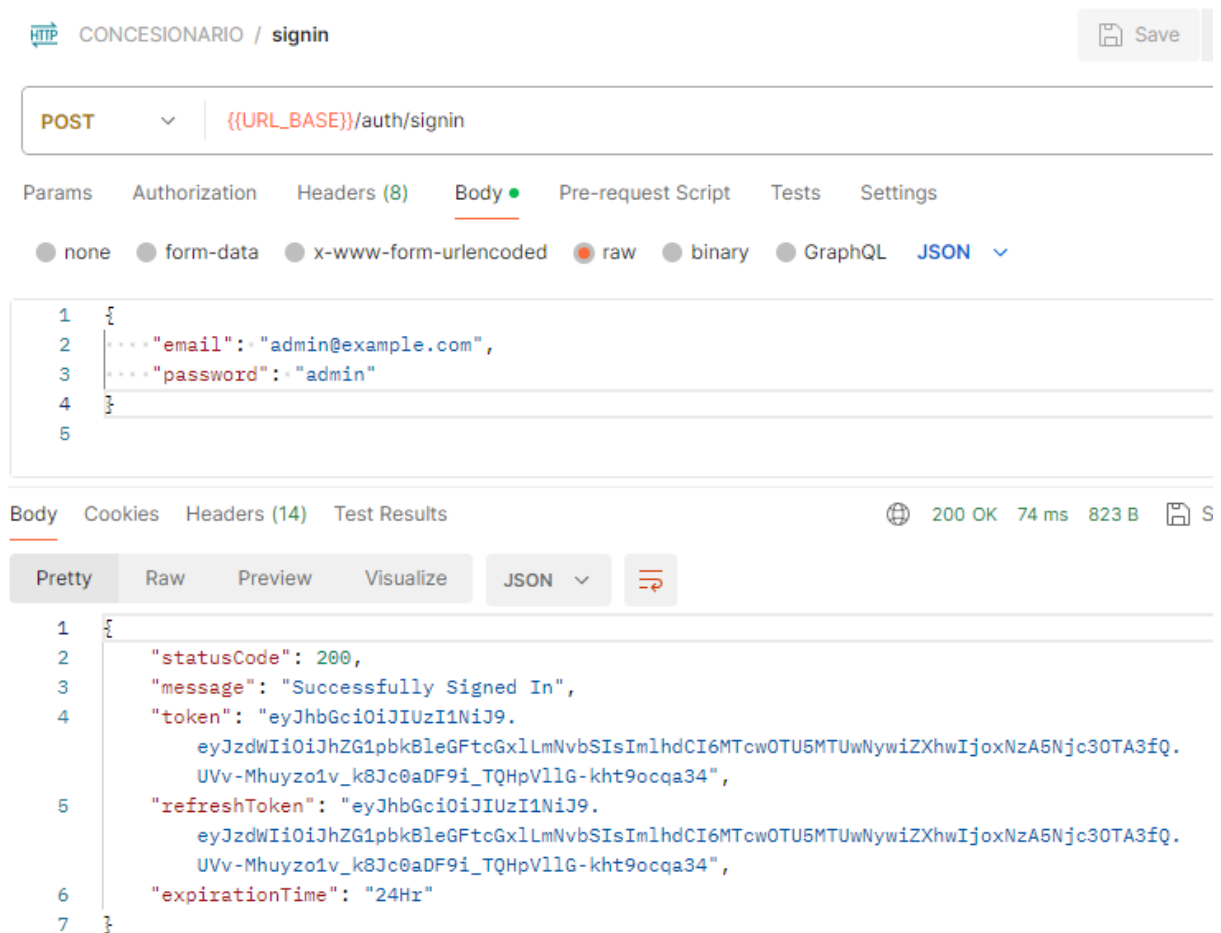
```
1 {
2   "statusCode": 200,
3   "message": "User Saved Successfully",
4   "ourUsers": {
5     "id": 6,
6     "email": "ejemplo1@ejemplo.com",
7     "password": "$2a$10$suJZETcI58s9uHRRMI2xQOG3oNKwSuIvaBwf10D/x0FOG6nGKIhEK",
8     "role": "USER",
9     "enabled": true,
10    "username": "ejemplo1@ejemplo.com",
11    "authorities": [
12      {
13        "authority": "USER"
14      }
15    ],
16    "credentialsNonExpired": true,
17    "accountNonExpired": true,
```

Por defecto, el usuario que se crea tiene rol de usuario normal, para que no tenga acceso al CRUD completo.

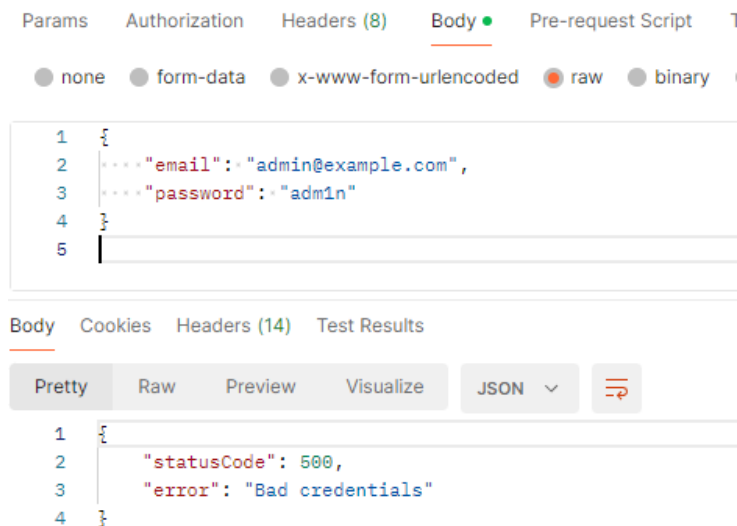
Iniciar sesión

Ya podemos iniciar sesión con esta cuenta:

Nos dirigimos al signin y hacemos un POST con el json del usuario creado en el body:



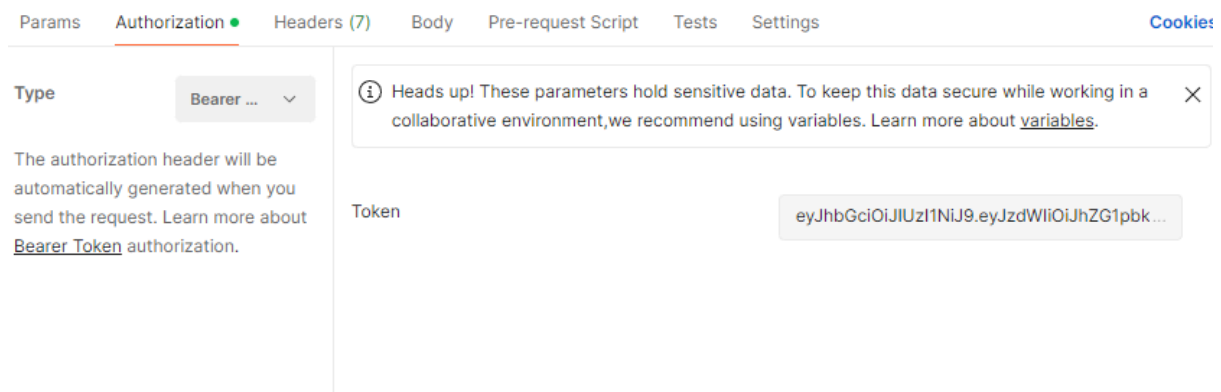
Nos devuelve el mensaje de que se ha iniciado sesión y nos da un token.



Si introducimos credenciales incorrectas, nos avisará. Para realizar los demás tests, iniciamos sesión con la cuenta del admin y guardamos el token.

Configuración tests

Para los siguientes tests, necesitamos configurar el token en el apartado de Autorización de cada test que vayamos a hacer:



1. Obtener coches

Nos dirigimos a "URL/coche" y realizamos un GET para recuperar todos los coches.

HTTP CONCESIONARIO / obtener coches

GET `{{URL_BASE}}/coche`

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Type Bearer Token

Heads up! These parameters hold sensitive data. To keep thi variables.

The authorization header will be automatically generated when

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 9,
4     "marca": "Bmw",
5     "modelo": "420d",
6     "anyo": 2019,
7     "potencia": 150,
8     "kilometraje": 35000,
9     "peso": 1500,
10    "combustible": "Gasolina",
11    "color": "Negro",
12    "precio": 40000,
13    "descripcion": "Coche deportivo"
14  },
15  {
16    "id": 10,
17    "marca": "Mercedes-Benz",
18    "modelo": "C200",
19    "anyo": 2018,
20    "potencia": 180,
21    "kilometraje": 40000,
22    "peso": 1600,
23    "combustible": "Gasolina",
24    "color": "Blanco",
25    "precio": 45000,
26    "descripcion": "Coche deportivo"
27  },
28  {
29    "id": 11,
30    "marca": "Ford",
31    "modelo": "Focus",
```

2. Obtener un coche por id

Nos dirigimos a "URL/coche/id" y realizamos un GET para recuperar el coche según el id.

HTTP CONCESIONARIO / obtener coche por id

GET {{URL_BASE}}/coche/10

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Type Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Heads up! These parameters are variables.

Token

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 10,
3   "marca": "Mercedes-Benz",
4   "modelo": "C200",
5   "anyo": 2018,
6   "potencia": 180,
7   "kilometraje": 40000,
8   "peso": 1600,
9   "combustible": "Gasolina",
10  "color": "Blanco",
11  "precio": 45000,
12  "descripcion": "Coche deportivo"
13 }
```

3. Obtener motores

Realizamos los mismos pasos que en el caso de los coches, pero cambiando coche por motor: "URL/motor"

HTTP CONCESIONARIO / obtener motores

GET {{URL_BASE}}/motor

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Type Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#)

Heads up! These parameters hold sensitive data. To keep [variables](#).

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON

```

1  [
2    {
3      "id": 5,
4      "nombre": "OM651",
5      "fabricante": "Mercedes-Benz",
6      "anyoFabricacion": 2019,
7      "tipoCombustible": "Gasolina",
8      "consumo": 5.5,
9      "vidaUtil": 400000
10   },
11   {
12     "id": 6,
13     "nombre": "B47",
14     "fabricante": "BMW",
15     "anyoFabricacion": 2018,
16     "tipoCombustible": "Gasolina",
17     "consumo": 6.0,
18     "vidaUtil": 350000
19   },
20   {
21     "id": 7,
22     "nombre": "K20C2",
23     "fabricante": "Honda",
24     "anyoFabricacion": 2020,
25     "tipoCombustible": "Gasolina",
26     "consumo": 7.5,
27     "vidaUtil": 250000
28   },
29   {
30     "id": 8,

```

4. Obtener un motor por id

Nos dirigimos a "URL/motor/id" y realizamos un GET para recuperar el motor según el id.

HTTP CONCESIONARIO / obtener motor por id

GET `{{URL_BASE}}/motor/5`

Params Authorization • Headers (7) Body Pre-request Script Tests Set

Type Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Heads up! These p: [variables](#).

Token

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 5,
3   "nombre": "OM651",
4   "fabricante": "Mercedes-Benz",
5   "anyoFabricacion": 2019,
6   "tipoCombustible": "Gasolina",
7   "consumo": 5.5,
8   "vidaUtil": 400000
9 }
```

5. Crear coche

Nos dirigimos a “URL/coche”, introducimos el token de un admin, pasamos el coche por body, y realizamos un POST.

HTTP CONCESIONARIO / crear coche

POST {{URL_BASE}}/coche

Params Authorization Headers (9) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "marca": "Mercedes-Benz",
3   "modelo": "S-Class",
4   "anyo": 2022,
5   "potencia": 469,
6   "kilometraje": 15000,
7   "peso": 5050,
8   "combustible": "Gasolina",
9   "color": "Negro",
10  "precio": 100000,
11  "descripcion": "Mercedes-Benz S-Class 2022, una obra maestra de lujo y rendimiento.
12                  generación y un rendimiento excepcional en carretera."
13 }

```

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "id": 18,
3   "marca": "Mercedes-Benz",
4   "modelo": "S-Class",
5   "anyo": 2022,
6   "potencia": 469,
7   "kilometraje": 15000,
8   "peso": 5050,
9   "combustible": "Gasolina",
10  "color": "Negro",
11  "precio": 100000,
12  "descripcion": "Mercedes-Benz S-Class 2022, una obra maestra de lujo y rendimient
13                  generación y un rendimiento excepcional en carretera."

```

6. Crear motor

Nos dirigimos a "URL/motor", introducimos el token de un admin, pasamos el motor por body, y realizamos un POST.

The screenshot shows a REST client interface with the following details:

- Method:** POST
- URL:** `{{URL_BASE}}/motor`
- Body Type:** raw (selected)
- Request Body:**

```
1 {
2   "nombre": "B48",
3   "fabricante": "BMW",
4   "anyoFabricacion": 2018,
5   "tipoCombustible": "Diesel",
6   "consumo": 6.0,
7   "vidaUtil": 350000
8 }
9
```
- Response Body:**

```
1 {
2   "id": 9,
3   "nombre": "B48",
4   "fabricante": "BMW",
5   "anyoFabricacion": 2018,
6   "tipoCombustible": "Diesel",
7   "consumo": 6.0,
8   "vidaUtil": 350000
9 }
```

7. Actualizar coche

Nos dirigimos a coche y tenemos que introducir el id del coche que queremos actualizar "URL/coche/id".

En este caso vamos a modificar el siguiente:

```
{
  "id": 18,
  "marca": "Mercedes-Benz",
  "modelo": "S-Class",
  "anyo": 2022,
  "potencia": 469,
  "kilometraje": 15000,
  "peso": 5050,
  "combustible": "Gasolina",
  "color": "Negro",
  "precio": 100000,
  "descripcion": "Mercedes-Benz S-Class 2022, una obra maestra d
    generación y un rendimiento excepcional en carretera."
}
```

Pasamos el coche actualizado por body y tiene que dar un status: 200 OK.

[HTTP](#) CONCESIONARIO / Actualizar coche

PUT

{{URL_BASE}}/coche/9

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```

1  {
2    "marca": "Mercedes-Benz ACTUALIZADO",
3    "modelo": "S-Class",
4    "anyo": 2023,
5    "potencia": 469,
6    "kilometraje": 15000,
7    "peso": 5050,
8    "combustible": "Gasolina",
9    "color": "Rojo",
10   "precio": 100000,
11   "descripcion": "Mercedes-Benz S-Class 2022, una obra maestra de lujo y rendimiento
    generación y un rendimiento excepcional en carretera."
12 }

```

Body

Cookies

Headers (14)

Test Results

Pretty

Raw

Preview

Visualize

JSON

```

1  {
2    "id": 9,
3    "marca": "Mercedes-Benz ACTUALIZADO",
4    "modelo": "S-Class",
5    "anyo": 2023,
6    "potencia": 150,
7    "kilometraje": 15000,
8    "peso": 5050,
9    "combustible": "Gasolina",
10   "color": "Rojo",
11   "precio": 100000,
12   "descripcion": "Mercedes-Benz S-Class 2022, una obra maestra de lujo y rendimien
    generación y un rendimiento excepcional en carretera."
13 }

```

8. Actualizar motor

Para actualizar un motor, nos dirigimos a motor y tenemos que introducir el id del motor que queremos actualizar "URL/motor/id".

En este caso vamos a modificar el siguiente:

```
{  
  "id": 9,  
  "nombre": "B48",  
  "fabricante": "BMW",  
  "anyoFabricacion": 2018,  
  "tipoCombustible": "Diesel",  
  "consumo": 6.0,  
  "vidaUtil": 350000  
}
```

Pasamos el motor actualizado por body y tiene que dar un status: 200 OK.

HTTP CONCESIONARIO / Actualizar motor

PUT {{URL_BASE}}/motor/9

Params Authorization Headers (9) Body Pre-request Script Tests

none form-data x-www-form-urlencoded raw binary GraphQL

```
1 {
2   "id": 9,
3   "nombre": "B48 ACTUALIZADO",
4   "fabricante": "BMW",
5   "anyoFabricacion": 2018,
6   "tipoCombustible": "Diesel",
7   "consumo": 6.0,
8   "vidaUtil": 300000
9 }
```

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 9,
3   "nombre": "B48 ACTUALIZADO",
4   "fabricante": "BMW",
5   "anyoFabricacion": 2018,
6   "tipoCombustible": "Diesel",
7   "consumo": 6.0,
8   "vidaUtil": 300000
9 }
```

9. Borrar coche

Para borrar un coche, nos dirigimos a coche y tenemos que introducir el id del coche que queremos borrar "URL/coche/id".

En este caso, vamos a borrar el que tiene id=9

HTTP CONCESIONARIO / obtener coches

GET {{URL_BASE}}/coche

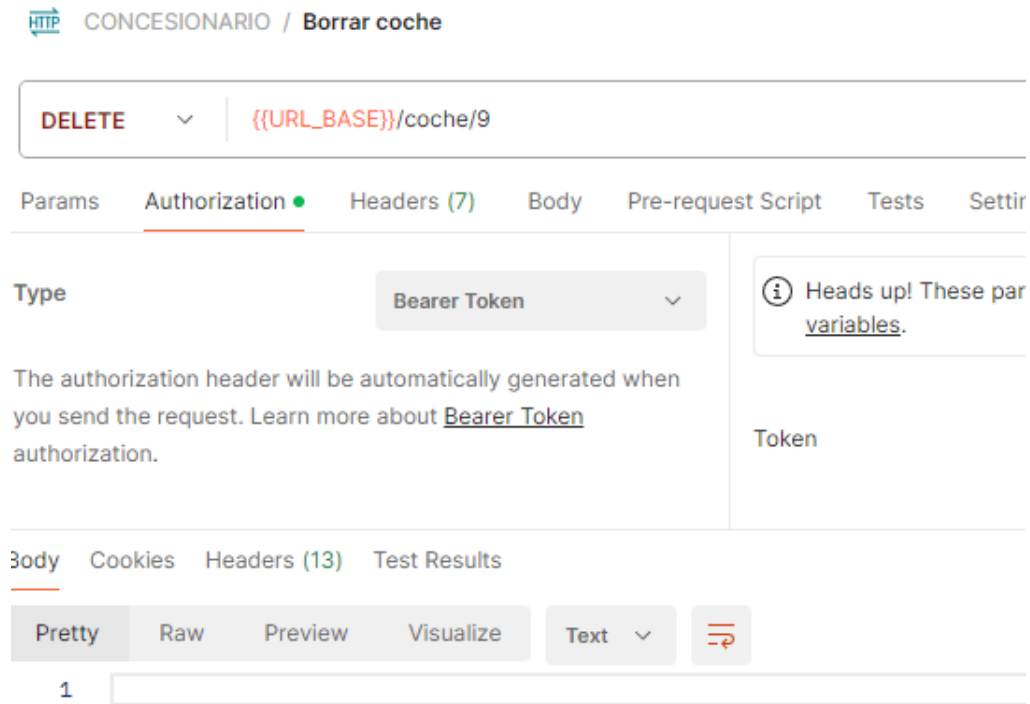
Params Authorization Headers (7) Body Pre-request Script Tests Setting

Type Bearer Token Heads up! These parameters are variables.

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 9,
4     "marca": "Mercedes-Benz ACTUALIZADO",
5     "modelo": "S-Class",
6     "anyo": 2023,
7     "potencia": 150,
8     "kilometraje": 15000,
9     "peso": 5050,
10    "combustible": "Gasolina",
11    "color": "Rojo",
12    "precio": 100000,
13    "descripcion": "Mercedes-Benz S-Class 2022, una obra maestra de
14                  generación y un rendimiento excepcional en carretera."
15  },
16  ]
```



Si nos devuelve un Status: 200 OK, es que se ha realizado correctamente.

Si volvemos a comprobar todos los coches, vemos que no aparece:

HTTP CONCESIONARIO / obtener coches

GET `{{URL_BASE}}/coche`

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Type Bearer Token ⌵ ⓘ Heads up! These parameters hold variables.

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON ⌵ ≡

```

1  [
2    {
3      "id": 10,
4      "marca": "Mercedes-Benz",
5      "modelo": "C200",
6      "anyo": 2018,
7      "potencia": 180,
8      "kilometraje": 40000,
9      "peso": 1600,
10     "combustible": "Gasolina",
11     "color": "Blanco",
12     "precio": 45000,
13     "descripcion": "Coche deportivo"
14   },
15 ]

```

10. Borrar motor

Para borrar un motor, nos dirigimos al motor y tenemos que introducir el id del motor que queremos borrar "URL/motor/id".

En este caso, vamos a borrar el que tiene id=5

HTTP CONCESIONARIO / obtener motores

GET {{URL_BASE}}/motor

Params Authorization Headers (7) Body Pre-request Script Tests Settings

Type Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Heads up! These parameters are variables.

Token

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 5,
4     "nombre": "OM651",
5     "fabricante": "Mercedes-Benz",
6     "anyoFabricacion": 2019,
7     "tipoCombustible": "Gasolina",
8     "consumo": 5.5,
9     "vidaUtil": 400000
10  },
```

The screenshot shows a REST client interface with the following components:

- Top Bar:** HTTP icon, "CONCESIONARIO / Borrar motor".
- Request Bar:** Method "DELETE" and URL "{{URL_BASE}}/motor/5".
- Authorization Tab:** Selected. Shows "Type" as "Bearer Token". A note states: "The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization." A warning box says: "Heads up! These variables." A "Token" input field is present.
- Body Tab:** Selected. Shows a "Pretty" view of the response body, which is currently empty.
- Response Bar:** Status: 200 OK, Time: 11 ms, Size: 382 B.

Si nos devuelve un Status: 200 OK, es que se ha realizado correctamente.

Si volvemos a comprobar todos los motores, vemos que no aparece:

HTTP CONCESIONARIO / obtener motores

GET {{URL_BASE}}/motor

Params Authorization Headers (7) Body Pre-request Script Tests S

Type Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Heads up! These variables.

Token

Body Cookies Headers (14) Test Results

Pretty Raw Preview Visualize JSON

```
1 [
2   {
3     "id": 6,
4     "nombre": "B47",
5     "fabricante": "BMW",
6     "anyoFabricacion": 2018,
7     "tipoCombustible": "Gasolina",
8     "consumo": 6.0,
9     "vidaUtil": 350000
10  }
```