

Manual de Usuario – Resto de CRUDS

Links:

Link de trello:

<https://trello.com/invite/b/698fad50855b9b94bb6d2d4f/ATTIdad0af1fa598c9f2efb59c4ea658a88a527BD15C/tarea-de-taller-semana-5>

Link del repositorio:

<https://github.com/KevinLancerio-2022063/RepuestosAutomotrices.git>

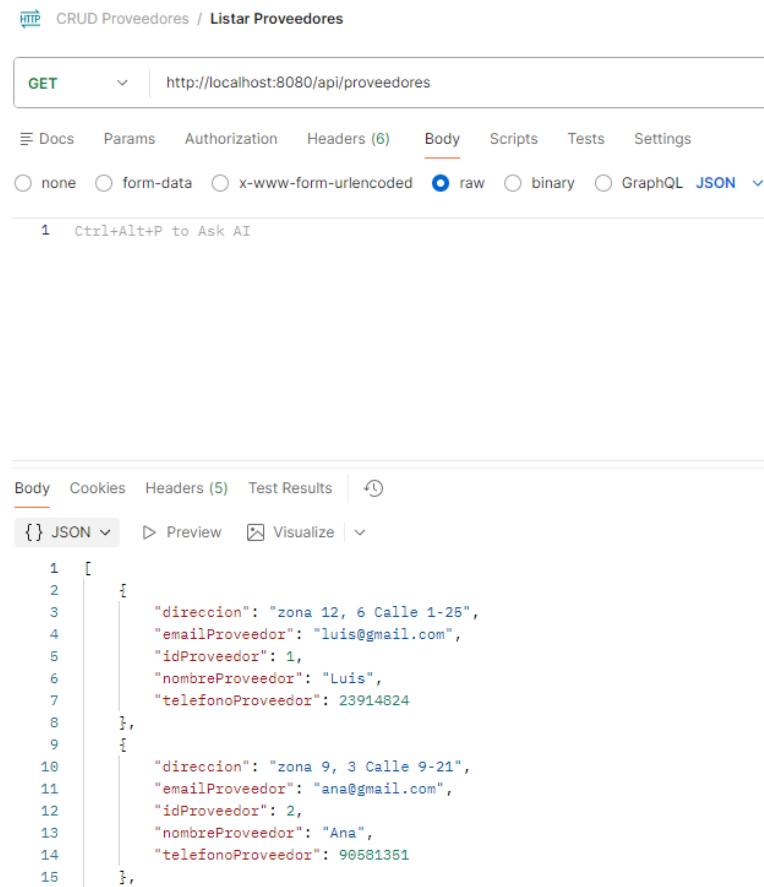
Link de Postman:

<https://kevinlancerio-2022063-4184979.postman.co/workspace/KevinLancerio-2022063's-Workspa-b7c2f8d8-7f21-4d32-86bd-1ceff9b114a3/request/52183058-f52bef0d-0433-417d-a85e-c9b9b987a6dc?action=share&creator=52183058>

CRUD Proveedores

Método GET - Listar

Con este método podemos ver a los proveedores que tenemos registrados.



The screenshot shows a Postman interface for a 'Listar Proveedores' request. The URL is set to `http://localhost:8080/api/proveedores`. The 'Body' tab is selected, showing a raw JSON response. The response is a list of two provider objects:

```
[{"idProveedor": 1, "nombreProveedor": "Luis", "telefonoProveedor": 23914824, "emailProveedor": "luis@gmail.com", "direccion": "zona 12, 6 Calle 1-25"}, {"idProveedor": 2, "nombreProveedor": "Ana", "telefonoProveedor": 90581351, "emailProveedor": "ana@gmail.com", "direccion": "zona 9, 3 Calle 9-21"}]
```

Lo podemos ver también gracias a este link en el navegador:
<http://localhost:8080/api/proveedores>

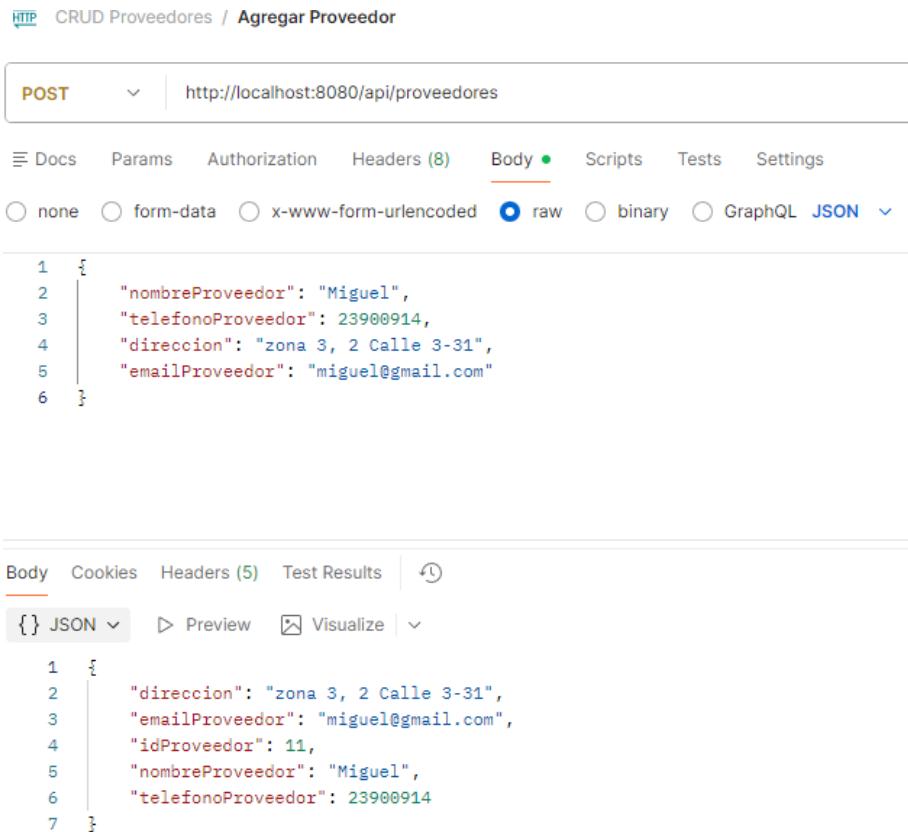


Dar formato al texto

```
[  
 {  
   "direccion": "zona 12, 6 Calle 1-25",  
   "emailProveedor": "luis@gmail.com",  
   "idProveedor": 1,  
   "nombreProveedor": "Luis",  
   "telefonoProveedor": 23914824  
 },  
 {  
   "direccion": "zona 9, 3 Calle 9-21",  
   "emailProveedor": "ana@gmail.com",  
   "idProveedor": 2,  
   "nombreProveedor": "Ana",  
   "telefonoProveedor": 90581351  
 },
```

Método POST - Agregar

Con este método podemos agregar el proveedor que queramos respetando la estructura de nuestra base de datos y verificamos que se haya agregado correctamente en la página web.



HTTP CRUD Proveedores / Agregar Proveedor

POST http://localhost:8080/api/proveedores

Docs Params Authorization Headers (8) Body Scripts Tests Settings

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

```
1 {  
2   "nombreProveedor": "Miguel",  
3   "telefonoProveedor": 23900914,  
4   "direccion": "zona 3, 2 Calle 3-31",  
5   "emailProveedor": "miguel@gmail.com"  
6 }
```

Body Cookies Headers (5) Test Results

{ } **JSON** Preview Visualize

```
1 {  
2   "direccion": "zona 3, 2 Calle 3-31",  
3   "emailProveedor": "miguel@gmail.com",  
4   "idProveedor": 11,  
5   "nombreProveedor": "Miguel",  
6   "telefonoProveedor": 23900914  
7 }
```

```
{
  "direccion": "zona 3, 2 Calle 3-31",
  "emailProveedor": "miguel@gmail.com",
  "idProveedor": 11,
  "nombreProveedor": "Miguel",
  "telefonoProveedor": 23900914
}
]
```

Método DELETE - Eliminar

Con este método eliminamos el proveedor que queramos colocando el id en la URL, luego podemos verificar que efectivamente se ha eliminado el proveedor.

HTTP CRUD Proveedores / Eliminar Proveedor

DELETE http://localhost:8080/api/proveedores/11

☰ Docs Params Authorization Headers (6) Body Scripts Tests Settings

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

1 Ctrl+Alt+P to Ask AI

Body Cookies Headers (5) Test Results

Raw Preview Visualize

1 Proveedor eliminado correctamente

```
{
  "direccion": "zona 10, 8 Calle 12-1",
  "emailProveedor": "fernando@gmail.com",
  "idProveedor": 10,
  "nombreProveedor": "Fernando",
  "telefonoProveedor": 69492550
}
```

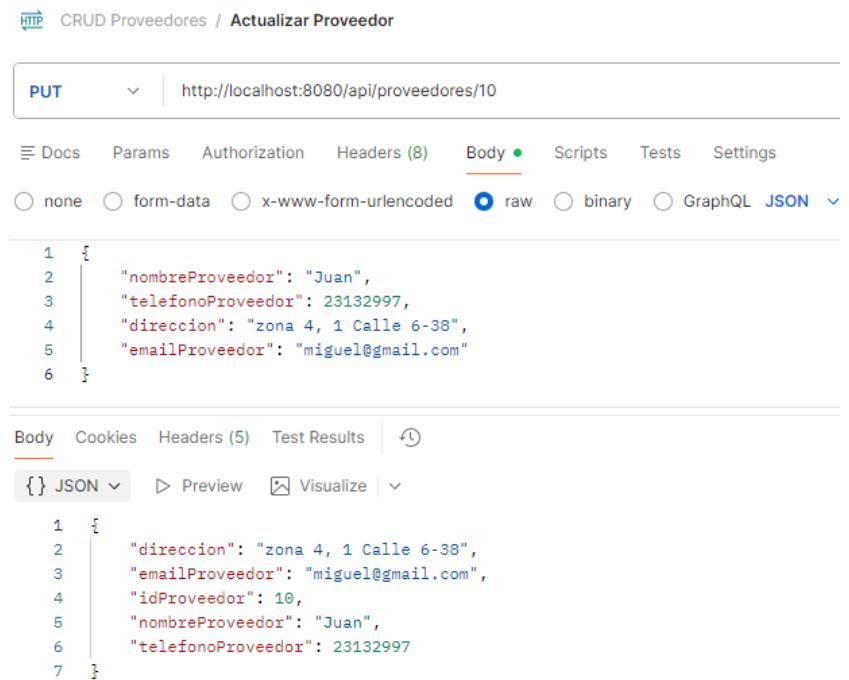
Método PUT - Eliminar

Con este método podemos actualizar el proveedor que queramos según su id respetando la estructura de nuestra base de datos de la entidad Proveedores, al igual que los otros métodos podemos asegurarnos de la información yendo a la página.

Antes:

```
{  
    "direccion": "zona 10, 8 Calle 12-1",  
    "emailProveedor": "fernando@gmail.com",  
    "idProveedor": 10,  
    "nombreProveedor": "Fernando",  
    "telefonoProveedor": 69492550  
}  
  
[  
  
{  
    "direccion": "zona 4, 1 Calle 6-38",  
    "emailProveedor": "miguel@gmail.com",  
    "idProveedor": 10,  
    "nombreProveedor": "Juan",  
    "telefonoProveedor": 23132997  
}  
]
```

Aplicamos el método



The screenshot shows the Postman interface with a PUT request to `http://localhost:8080/api/proveedores/10`. The request body contains the following JSON:

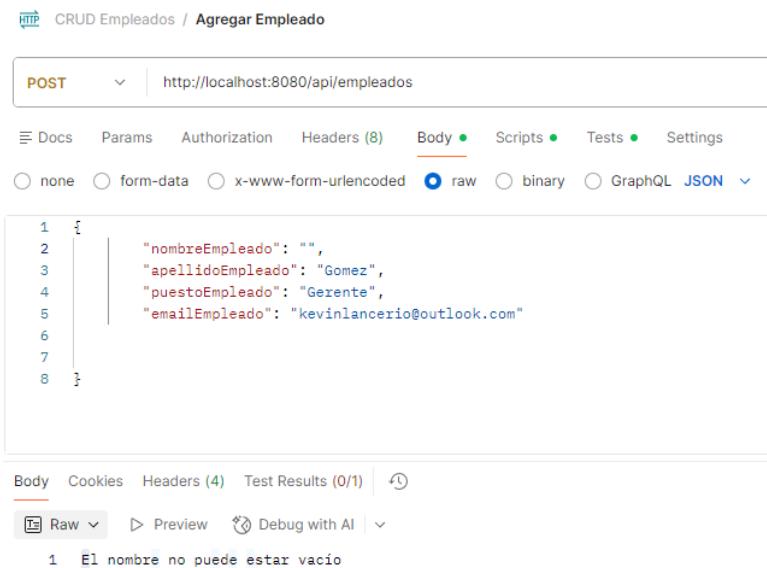
```
1 {  
2     "nombreProveedor": "Juan",  
3     "telefonoProveedor": 23132997,  
4     "direccion": "zona 4, 1 Calle 6-38",  
5     "emailProveedor": "miguel@gmail.com"  
6 }
```

The response body shows the updated supplier information:

```
1 {  
2     "direccion": "zona 4, 1 Calle 6-38",  
3     "emailProveedor": "miguel@gmail.com",  
4     "idProveedor": 10,  
5     "nombreProveedor": "Juan",  
6     "telefonoProveedor": 23132997  
7 }
```

Validaciones

Se agregaron validaciones como que el nombre no puede estar vacío:



The screenshot shows the Postman interface with a POST request to `http://localhost:8080/api/empleados`. The request body contains the following JSON:

```
1 {  
2     "nombreEmpleado": "",  
3     "apellidoEmpleado": "Gomez",  
4     "puestoEmpleado": "Gerente",  
5     "emailEmpleado": "kevinalancerio@outlook.com"  
6  
7  
8 }
```

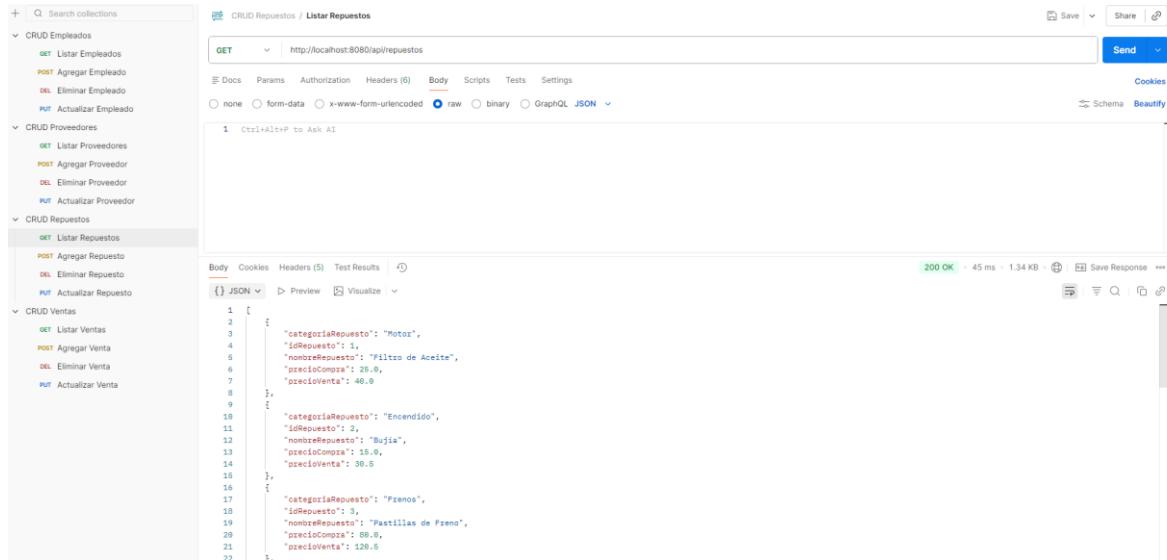
The response body shows an error message:

```
1 El nombre no puede estar vacio
```

CRUD Repuestos

Método GET - Listar

Con este método listamos/vemos a todos los repuestos que tengamos en la base de datos con el call en los procedimientos almacenados.



The screenshot shows the Postman interface with the following details:

- Collection:** CRUD Repuestos
- Request:** GET /api/repuestos
- Body:** raw JSON (selected)
- Response:** 200 OK (45 ms, 1.34 KB)
- JSON Response:**

```
[{"idRepuesto": 1, "nombreRepuesto": "Filtro de Aceite", "categoriaRepuesto": "Motor", "precioCompra": 25, "precioVenta": 40}, {"idRepuesto": 2, "nombreRepuesto": "Bujía", "categoriaRepuesto": "Encendido", "precioCompra": 15, "precioVenta": 30.5}, {"idRepuesto": 3, "nombreRepuesto": "Pastillas de Freno", "categoriaRepuesto": "Frenos", "precioCompra": 80, "precioVenta": 120.5}]
```

Hay otra forma de ver si esto funciona y es colocar este link en el navegador <http://localhost:8080/api/repuestos> y nos debería salir esto:



The screenshot shows a browser window with the following details:

- Address Bar:** localhost:8080/api/repuestos
- Text Input:** Dar formato al texto
- JSON Response:**

```
[{"idRepuesto": 1, "nombreRepuesto": "Filtro de Aceite", "categoriaRepuesto": "Motor", "precioCompra": 25, "precioVenta": 40}, {"idRepuesto": 2, "nombreRepuesto": "Bujía", "categoriaRepuesto": "Encendido", "precioCompra": 15, "precioVenta": 30.5}, {"idRepuesto": 3, "nombreRepuesto": "Pastillas de Freno", "categoriaRepuesto": "Frenos", "precioCompra": 80, "precioVenta": 120.5}]
```

Además, en nuestra aplicación de IntelliJ nos debería salir esto

```
Hibernate: select r1_0.id_reuesto,r1_0.categoría_reuesto,r1_0.nombre_reuesto,r1_0.precio_compra,r1_0.precio_venta from repuestos r1_0
Hibernate: select r1_0.id_reuesto,r1_0.categoría_reuesto,r1_0.nombre_reuesto,r1_0.precio_compra,r1_0.precio_venta from repuestos r1_0
```

Método POST - Agregar

Con este método agregamos un repuesto que tenga los mismos datos que los demás repuestos menos el id porque es auto incrementable, para agregar un repuesto tenemos que colocar los datos en el mismo orden que aparece en nuestra entidad de la base de datos, como hay un FK, lo agregamos a la clase Repuestos y así podemos agregar correctamente.

The screenshot shows a Postman interface for a POST request to `http://localhost:8080/api/repuestos`. The request body is set to raw JSON:

```
1 {
2     "nombreRepuesto": "Filtro de aceite",
3     "categoriaRepuesto": "Mantenimiento",
4     "precioCompra": 15.5,
5     "precioVenta": 25.00,
6     "proveedor": {
7         "idProveedor": 1
8     }
9 }
```

The response body shows the newly created part with its ID filled in:

```
1 {
2     "categoriaRepuesto": "Mantenimiento",
3     "idRepuesto": 14,
4     "nombreRepuesto": "Filtro de aceite",
5     "precioCompra": 15.5,
6     "precioVenta": 25.0,
7     "proveedor": {
8         "direccion": null,
9         "emailProveedor": null,
10        "idProveedor": 1,
11        "nombreProveedor": null,
12        "telefonoProveedor": null
13    }
14 }
```

Y así aparece en la página:

```
{  
    "categoriaRepuesto": "Mantenimiento",  
    "idRepuesto": 14,  
    "nombreRepuesto": "Filtro de aceite",  
    "precioCompra": 15.5,  
    "precioVenta": 25,  
    "proveedor": {  
        "direccion": "zona 12, 6 Calle 1-25",  
        "emailProveedor": "luis@gmail.com",  
        "idProveedor": 1,  
        "nombreProveedor": "Luis",  
        "telefonoProveedor": 23914824  
    }  
}  
]
```

Método DELETE - Eliminar

Con este método podemos eliminar el repuesto que queramos utilizando el id, solo hay que especificarlo en la URL para eliminarlo correctamente:

The screenshot shows the Postman interface with the following details:

- Method:** DELETE
- URL:** http://localhost:8080/api/repuestos/14
- Headers:** (6) (highlighted)
- Body:** raw (selected)
- JSON Response:** (highlighted)
- Body Content:** 1 Proveedor eliminado correctamente

Método PUT - Actualizar

Con este método podemos actualizar la información que queramos indicándolo con el id en la URL.

Antes:

```
{  
  "categoriaRepuesto": "Mantenimiento",  
  "idRepuesto": 12,  
  "nombreRepuesto": "Filtro de aceite",  
  "precioCompra": 15.5,  
  "precioVenta": 25,  
  "proveedor": {  
    "direccion": "zona 12, 6 Calle 1-25",  
    "emailProveedor": "luis@gmail.com",  
    "idProveedor": 1,  
    "nombreProveedor": "Luis",  
    "telefonoProveedor": 23914824  
  }  
}
```

Después:

```
{  
  "categoriaRepuesto": "Eléctrico",  
  "idRepuesto": 12,  
  "nombreRepuesto": "Batería 12V",  
  "precioCompra": 45.5,  
  "precioVenta": 55,  
  "proveedor": {  
    "direccion": "zona 12, 6 Calle 1-25",  
    "emailProveedor": "luis@gmail.com",  
    "idProveedor": 1,  
    "nombreProveedor": "Luis",  
    "telefonoProveedor": 23914824  
  }  
}
```

Hacemos el método

The screenshot shows the POSTMAN interface for a PUT request to update a car part. The URL is `http://localhost:8080/api/repuestos/12`. The request method is `PUT`. The `Body` tab is selected, showing a JSON payload with the updated values:

```
1 {  
2   "nombreRepuesto": "Batería 12V",  
3   "categoriaRepuesto": "Eléctrico",  
4   "precioCompra": 45.5,  
5   "precioVenta": 55.0,  
6   "proveedor": {  
7     "idProveedor": 1  
8   }  
9 }
```

Below the request, the response body is shown in JSON format, reflecting the updated data from the previous code snippet:

```
1 {  
2   "categoriaRepuesto": "Eléctrico",  
3   "idRepuesto": 12,  
4   "nombreRepuesto": "Batería 12V",  
5   "precioCompra": 45.5,  
6   "precioVenta": 55.0,  
7   "proveedor": {  
8     "direccion": "zona 12, 6 Calle 1-25",  
9     "emailProveedor": "luis@gmail.com",  
10    "idProveedor": 1,  
11    "nombreProveedor": "Luis",  
12    "telefonoProveedor": 23914824  
13  }  
14 }
```

CRUD Ventas

Método GET - Listar

Listamos todos los registros que hicimos en la base de datos con sus respectivos FK.

HTTP CRUD Ventas / Listar Ventas

GET http://localhost:8080/api/ventas

☰ Docs Params Authorization Headers (6) Body Scripts Tests Settings

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

1 Ctrl+Alt+P to Ask AI

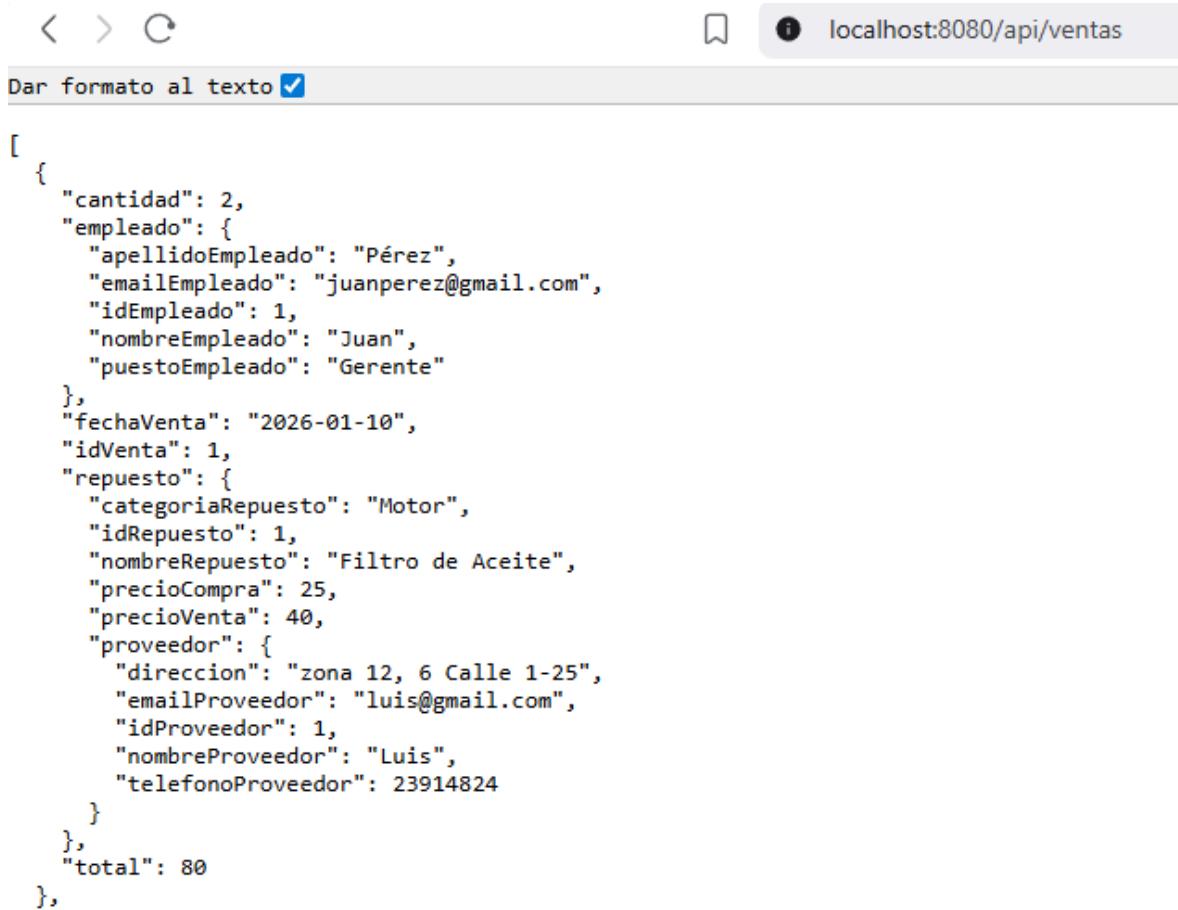
Body Cookies Headers (5) Test Results

{ } JSON Preview Visualize

```
1 [  
2 {  
3     "cantidad": 2,  
4     "empleado": {  
5         "apellidoEmpleado": "Pérez",  
6         "emailEmpleado": "juanperez@gmail.com",  
7         "idEmpleado": 1,  
8         "nombreEmpleado": "Juan",  
9         "puestoEmpleado": "Gerente"  
10    },  
11    "fechaVenta": "2026-01-10",  
12    "idVenta": 1,  
13    "repuesto": {  
14        "categoriaRepuesto": "Motor",  
15        "idRepuesto": 1,  
16        "nombreRepuesto": "Filtro de Aceite",  
17        "precioCompra": 25.0,  
18        "precioVenta": 40.0,  
19        "proveedor": {  
20            "direccion": "zona 12, 6 Calle 1-25",  
21            "emailProveedor": "luis@gmail.com",  
22            "idProveedor": 1,  
23            "nombreProveedor": "Luis",  
24            "telefonoProveedor": 23914824  
25        },  
26    },  
27    "total": 80.0  
28},
```

Para comprobar colocamos esto en el navegador: <http://localhost:8080/api/ventas>

Y nos tendría que salir esto.



A screenshot of a web browser window. The address bar shows "localhost:8080/api/ventas". The page content displays a JSON object representing a sale record:

```
[{"cantidad": 2, "empleado": {"apellidoEmpleado": "Pérez", "emailEmpleado": "juanperez@gmail.com", "idEmpleado": 1, "nombreEmpleado": "Juan", "puestoEmpleado": "Gerente"}, "fechaVenta": "2026-01-10", "idVenta": 1, "repuesto": {"categoriaRepuesto": "Motor", "idRepuesto": 1, "nombreRepuesto": "Filtro de Aceite", "precioCompra": 25, "precioVenta": 40, "proveedor": {"direccion": "zona 12, 6 Calle 1-25", "emailProveedor": "luis@gmail.com", "idProveedor": 1, "nombreProveedor": "Luis", "telefonoProveedor": 23914824}}, "total": 80}]
```

Y así nos tendría que quedar en el intelliJ

```
Test API
2026-02-13T21:29:45.468+06:00 INFO 19656 --- [Ejemplo] [nio-8080-exec-3] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring DispatcherServlet 'dispatcherServlet'
2026-02-13T21:29:45.468+06:00 INFO 19656 --- [Ejemplo] [nio-8080-exec-3] o.s.web.servlet.DispatcherServlet : Initializing Servlet 'dispatcherServlet'
2026-02-13T21:29:45.469+06:00 INFO 19656 --- [Ejemplo] [nio-8080-exec-3] o.s.web.servlet.DispatcherServlet : Completed initialization in 1 ms
Hibernate: select v1_0.id_venta, v1_0.cantidad, v1_0.id_empleado, v1_0.fecha_venta, v1_0.id_repuesto, v1_0.total from ventas v1_0
Hibernate: select e1_0.id_empleado, e1_0.apellido_empleado, e1_0.email_empleado, e1_0.nombre_empleado, e1_0.puesto_empleado from empleados e1_0 where e1_0.id_empleado=?
Hibernate: select r1_0.id_repuesto, r1_0.categoría_repuesto, r1_0.nombre_repuesto, r1_0.precio_compra, r1_0.precio_venta, r1_0.id_proveedor, p1_0.id_proveedor, p1_0.direccion, p1_0.email_proveedor, p1_0.nombre_proveedor, p1_0.telefono
Hibernate: select e1_0.id_empleado, e1_0.apellido_empleado, e1_0.email_empleado, e1_0.nombre_empleado, e1_0.puesto_empleado from empleados e1_0 where e1_0.id_empleado=?
Hibernate: select r1_0.id_repuesto, r1_0.categoría_repuesto, r1_0.nombre_repuesto, r1_0.precio_compra, r1_0.precio_venta, r1_0.id_proveedor, p1_0.id_proveedor, p1_0.direccion, p1_0.email_proveedor, p1_0.nombre_proveedor, p1_0.telefono
Hibernate: select e1_0.id_empleado, e1_0.apellido_empleado, e1_0.email_empleado, e1_0.nombre_empleado, e1_0.puesto_empleado from empleados e1_0 where e1_0.id_empleado=?
Hibernate: select r1_0.id_repuesto, r1_0.categoría_repuesto, r1_0.nombre_repuesto, r1_0.precio_compra, r1_0.precio_venta, r1_0.id_proveedor, p1_0.id_proveedor, p1_0.direccion, p1_0.email_proveedor, p1_0.nombre_proveedor, p1_0.telefono
Hibernate: select e1_0.id_empleado, e1_0.apellido_empleado, e1_0.email_empleado, e1_0.nombre_empleado, e1_0.puesto_empleado from empleados e1_0 where e1_0.id_empleado=?
```

Método POST - Agregar

Con este método agregamos una venta con sus datos correspondientes, agregamos también los FK que lleva esta entidad y así nos debería de quedar.

HTTP CRUD Ventas / Agregar Venta

POST http://localhost:8080/api/ventas

Docs Params Authorization Headers (8) **Body** Scripts Tests Settings

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

```
1 {  
2   "fechaVenta": "2026-02-13",  
3   "cantidad": 2,  
4   "total": 50.00,  
5   "empleado": {  
6     "idEmpleado": 1  
7   },  
8   "repuesto": {  
9     "idRepuesto": 1  
10  }  
11 }
```

Body Cookies Headers (5) Test Results

{ } JSON Preview Visualize

```
1 {  
2   "cantidad": 2,  
3   "empleado": {  
4     "apellidoEmpleado": null,  
5     "emailEmpleado": null,  
6     "idEmpleado": 1,  
7     "nombreEmpleado": null,  
8     "puestoEmpleado": null  
9   },  
10  "fechaVenta": "2026-02-13",  
11  "idVenta": 11,  
12  "repuesto": {  
13    "categoriaRepuesto": null,  
14    "idRepuesto": 1,  
15    "nombreRepuesto": null,  
16    "precioCompra": 0.0,  
17    "precioVenta": 0.0,  
18    "proveedor": null  
19  },  
20  "total": 50.0  
21 }
```

```
{
  "cantidad": 2,
  "empleado": {
    "apellidoEmpleado": "Pérez",
    "emailEmpleado": "juanperez@gmail.com",
    "idEmpleado": 1,
    "nombreEmpleado": "Juan",
    "puestoEmpleado": "Gerente"
  },
  "fechaVenta": "2026-02-13",
  "idVenta": 11,
  "repuesto": {
    "categoriaRepuesto": "Motor",
    "idRepuesto": 1,
    "nombreRepuesto": "Filtro de Aceite",
    "precioCompra": 25,
    "precioVenta": 40,
    "proveedor": {
      "direccion": "zona 12, 6 Calle 1-25",
      "emailProveedor": "luis@gmail.com",
      "idProveedor": 1,
      "nombreProveedor": "Luis",
      "telefonoProveedor": 23914824
    }
  },
  "total": 50
}
]
```

Método DELETE – Eliminar

Eliminamos la venta que tenga el id 11 y ya no nos aparece ni en la base de datos ni en la página.

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

- HTTP Method:** DELETE
- URL:** http://localhost:8080/api/ventas/11
- Headers:** (6) (highlighted in green)
- Body:** (radio buttons for none, form-data, x-www-form-urlencoded)
- Test Results:** (highlighted in green)
- Body Content:** 1 Venta eliminada correctamente

```
{
  "cantidad": 3,
  "empleado": {
    "apellidoEmpleado": "Morales",
    "emailEmpleado": "miguelmorales@gmail.com",
    "idEmpleado": 10,
    "nombreEmpleado": "Miguel",
    "puestoEmpleado": "Gerente"
  },
  "fechaVenta": "2026-01-19",
  "idVenta": 10,
  "repuesto": {
    "categoriaRepuesto": "Motor",
    "idRepuesto": 10,
    "nombreRepuesto": "Filtro de Aire",
    "precioCompra": 30.5,
    "precioVenta": 50,
    "proveedor": {
      "direccion": "zona 10, 8 Calle 12-1",
      "emailProveedor": "fernando@gmail.com",
      "idProveedor": 10,
      "nombreProveedor": "Fernanda",
      "telefonoProveedor": 69492550
    }
  },
  "total": 150
}
```

Método PUT - Actualizar

Con esto actualizamos la venta que tenga el id 10 y lo verificamos a través de la página.

The screenshot shows a POSTMAN interface with the following details:

- Method:** PUT
- URL:** <http://localhost:8080/api/ventas/10>
- Body (JSON):**

```

1  {
2    "fechaVenta": "2025-10-03",
3    "cantidad": 4,
4    "total": 880.00,
5    "empleado": {
6      "idEmpleado": 3
7    },
8    "repuesto": {
9      "idRepuesto": 4
10   }
11 }
```
- Response Body (JSON):**

```

1  {
2    "cantidad": 4,
3    "empleado": {
4      "apellidoEmpleado": null,
5      "emailEmpleado": null,
6      "idEmpleado": 3,
7      "nombreEmpleado": null,
8      "puestoEmpleado": null
9    },
10   "fechaVenta": "2025-10-03",
11   "idVenta": 10,
12   "repuesto": {
13     "categoriaRepuesto": null,
14     "idRepuesto": 4,
15     "nombreRepuesto": null,
16     "precioCompra": 0.0,
17     "precioVenta": 0.0,
18     "proveedor": null
19   },
20   "total": 880.0
21 }
```

```
{
    "cantidad": 4,
    "empleado": {
        "apellidoEmpleado": "Vásquez",
        "emailEmpleado": "carlosvasquez@gmail.com",
        "idEmpleado": 3,
        "nombreEmpleado": "Carlos",
        "puestoEmpleado": "Vendedor"
    },
    "fechaVenta": "2025-10-03",
    "idVenta": 10,
    "repuesto": {
        "categoriaRepuesto": "Suspensión",
        "idRepuesto": 4,
        "nombreRepuesto": "Amortiguador",
        "precioCompra": 150,
        "precioVenta": 220,
        "proveedor": {
            "direccion": "zona 2, 2 Calle 6-20",
            "emailProveedor": "maria@gmail.com",
            "idProveedor": 4,
            "nombreProveedor": "María",
            "telefonoProveedor": 71074958
        }
    },
    "total": 880
}
]
```