

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/tree/develop>

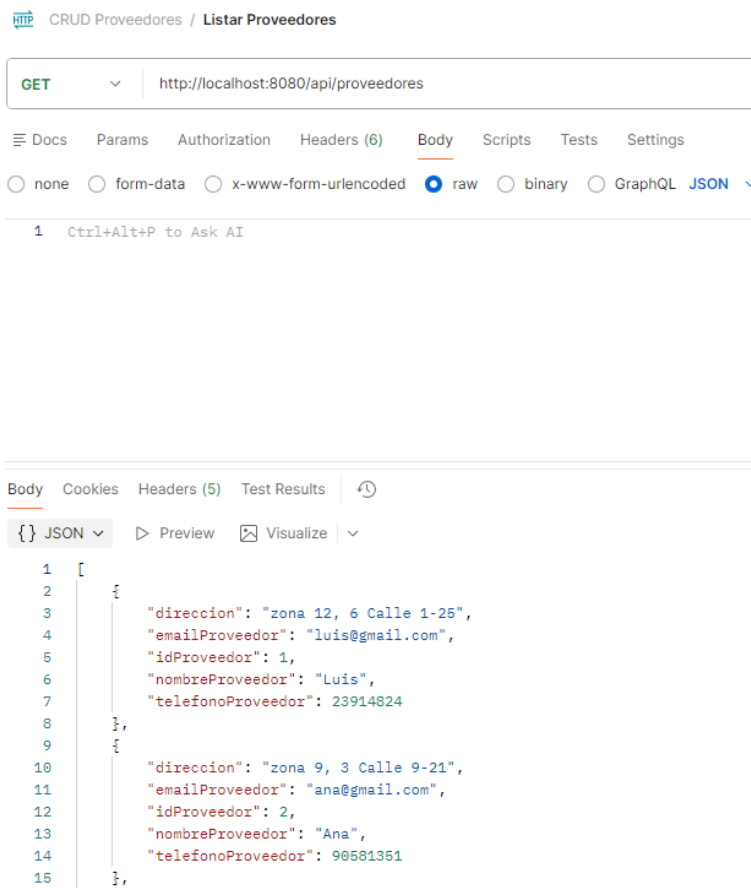
Link de Postman:

<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.



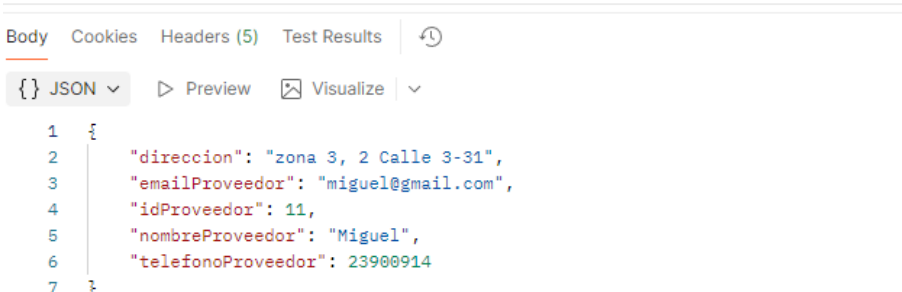
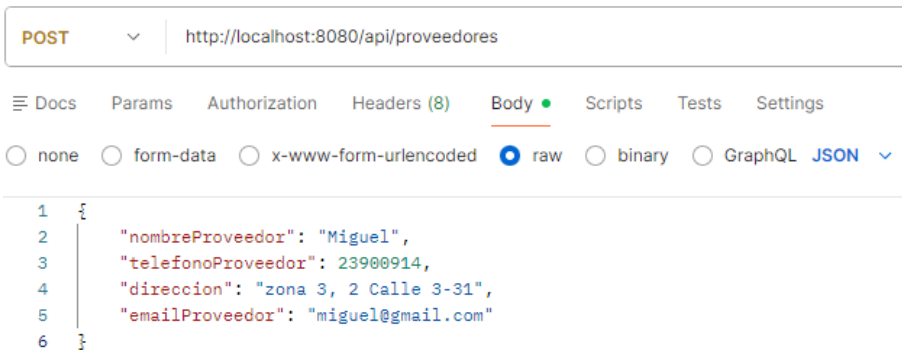
Lo podemos ver también gracias a este link en el navegador:
<http://localhost:8080/api/proveedores>



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



```
{
  "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.

 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 a REST client interface for a PUT request. The URL is `http://localhost:8080/api/proveedores/10`. The request body is a JSON object with the following fields: `nombreProveedor` (Juan), `telefonoProveedor` (23132997), `direccion` (zona 4, 1 Calle 6-38), and `emailProveedor` (miguel@gmail.com). The response body is a JSON object with the following fields: `direccion` (zona 4, 1 Calle 6-38), `emailProveedor` (miguel@gmail.com), `idProveedor` (10), `nombreProveedor` (Juan), and `telefonoProveedor` (23132997).

Validaciones

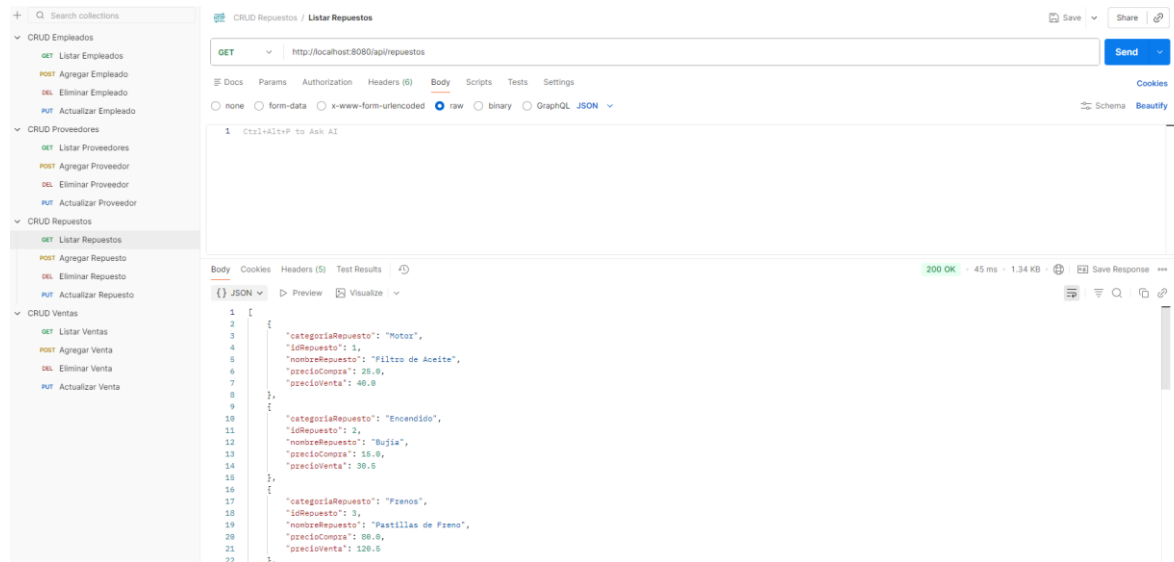
Se agregaron validaciones como que el nombre no puede estar vacío:

The screenshot shows a REST client interface for a POST request. The URL is `http://localhost:8080/api/empleados`. The request body is a JSON object with the following fields: `nombreEmpleado` (empty string), `apellidoEmpleado` (Gomez), `puestoEmpleado` (Gerente), and `emailEmpleado` (kevinlancerio@outlook.com). The response body is a validation error message: `El nombre no puede estar vacío`.

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.



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:



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

Además, en nuestra aplicación de IntelliJ nos debería salir esto

```
Hibernate: select r1_0.id_repuesto,r1_0.categoria_repuesto,r1_0.nombre_repuesto,r1_0.precio_compra,r1_0.precio_venta from repuestos r1_0
Hibernate: select r1_0.id_repuesto,r1_0.categoria_repuesto,r1_0.nombre_repuesto,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.

 CRUD Repuestos / Agregar Repuesto

POST

http://localhost:8080/api/repuestos

Docs

Params

Authorization

Headers (8)

Body

Scripts

Tests

Settings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

```
1  {
2    "nombreRepuesto": "Filtro de aceite",
3    "categoriaRepuesto": "Mantenimiento",
4    "precioCompra": 15.50,
5    "precioVenta": 25.00,
6    "proveedor": {
7      "idProveedor": 1
8    }
9  }
```

Body

Cookies

Headers (5)

Test Results

JSON

Preview

Visualize


```
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:


 **CRUD Repuestos / Eliminar Repuesto**






DELETE  `http://localhost:8080/api/repuestos/14`

 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

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

CRUD Repuestos / Actualizar Repuesto

PUT http://localhost:8080/api/repuestos/12

Docs Params Authorization Headers (8) Body Scripts Tests Settings

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

```
1 {
2   "nombreRepuesto": "Batería 12V",
3   "categoriaRepuesto": "Eléctrico",
4   "precioCompra": 45.50,
5   "precioVenta": 55.00,
6   "proveedor": {
7     "idProveedor": 1
8   }
9 }
```

Body Cookies Headers (5) Test Results


{ } JSON Preview Visualize


```
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.





 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.

```
< > ↻ ⓘ localhost:8080/api/ventas
Dar formato al texto ☒

[
  {
    "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.␣.[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.categoria_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 from repuestos r1_0 join proveedores p1_0 on r1_0.id_proveedor=p1_0.id_proveedor where r1_0.id_repuesto=?
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.categoria_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 from repuestos r1_0 join proveedores p1_0 on r1_0.id_proveedor=p1_0.id_proveedor where r1_0.id_repuesto=?
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.

 CRUD Ventas / Eliminar Venta



DELETE
▼
http://localhost:8080/api/ventas/11

☰ Docs
Params
Authorization
Headers (6)
Bc

☐ none
 ☐ form-data
 ☐ x-www-form-urlencoded
 

1 Ctrl+Alt+P to Ask AI

Body
Cookies
Headers (5)
Test Results
↺

 Raw
▼
▶ Preview
 Visualize
▼

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": "Fernando",
      "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.

[HTTP](#) CRUD Ventas / Actualizar Venta

PUT

▼

http://localhost:8080/api/ventas/10

☰ Docs

Params

Authorization

Headers (8)

Body ●

Scripts

Tests

Settings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

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 }
```

Body

Cookies

Headers (5)

Test Results

🔄

{ } JSON ▼

▶ Preview

🖼 Visualize

▼

```
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
}
```