

Parcial 2

Brayhan Alejandro Ortiz Ballesteros

NRC: 863617

Docente

Ing. William Matallana Porras

Corporación universitaria Minuto de Dios

Ingeniería de sistemas

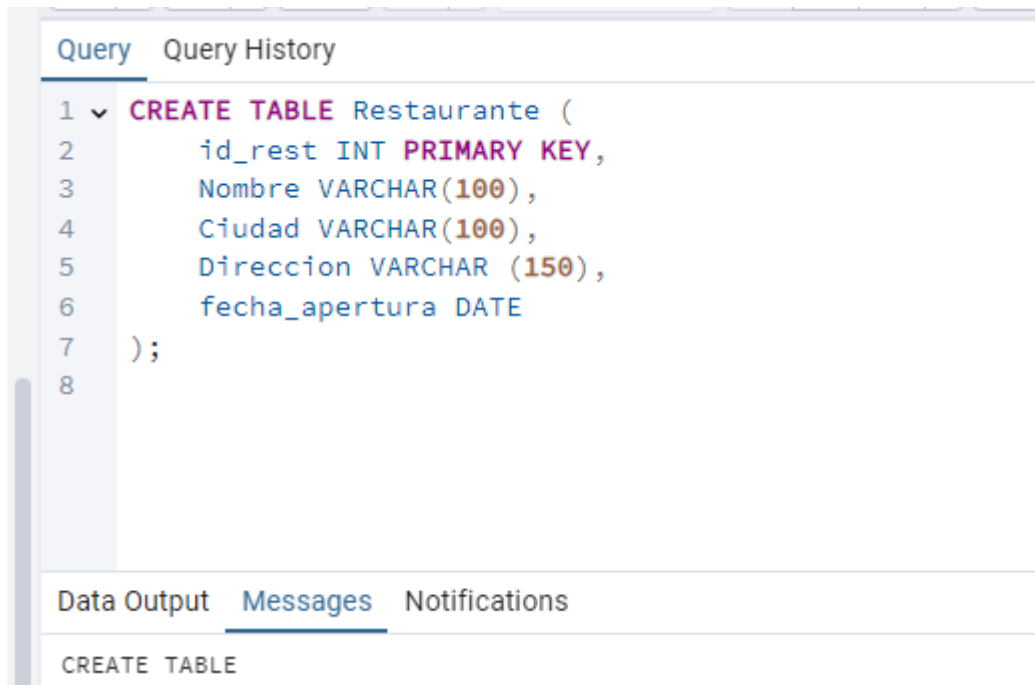
Zipaquirá

2025

Tabla de contenido

Crear tablas y Relaciones	3
Conexión a base de datos PostgreSQL (Supabase).	5
Crear rutas y controladores con Express para cada entidad (CRUD).....	6
Restaurante	6
Producto	8
Pedido	11
Detalles de pedido	13
Empleado	16
Implementar las siguientes consultas nativas y exponerlas por rutas:	18

Crear tablas y Relaciones



```
CREATE TABLE Restaurante (  
    id_rest INT PRIMARY KEY,  
    Nombre VARCHAR(100),  
    Ciudad VARCHAR(100),  
    Direccion VARCHAR (150),  
    fecha_apertura DATE  
);
```

```
CREATE TABLE Producto (  
    id_prod INT PRIMARY KEY,  
    Nombre VARCHAR(100),  
    Precio NUMERIC(10,2)  
);
```

```
CREATE TABLE Empleado (  
    id_empleado INT PRIMARY KEY,
```

```

Nombre VARCHAR(100),

Rol VARCHAR (50),

        id_rest INT, FOREIGN KEY (id_rest) REFERENCES restaurante(id_rest)

);

CREATE TABLE Pedido (

        id_pedido INT PRIMARY KEY,

        fecha DATE,

        total NUMERIC (10,2),

        id_rest INT, FOREIGN KEY (id_rest) REFERENCES restaurante(id_rest)

);

CREATE TABLE DetallePedido (

        id_detalle INT PRIMARY KEY,

        Cantidad INT,

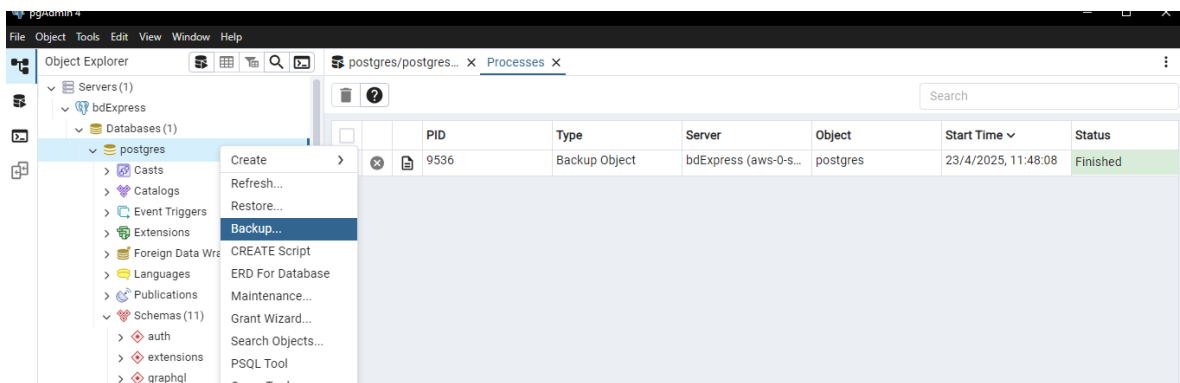
        Subtotal NUMERIC (10,2),

        id_pedido INT, FOREIGN KEY (id_pedido) REFERENCES Pedido(id_pedido),

        id_prod INT, FOREIGN KEY (id_prod) REFERENCES Producto(id_prod)

);

```



```

INSERT INTO Restaurante (id_rest, Nombre, Ciudad, Direccion, fecha_apertura) VALUES

```

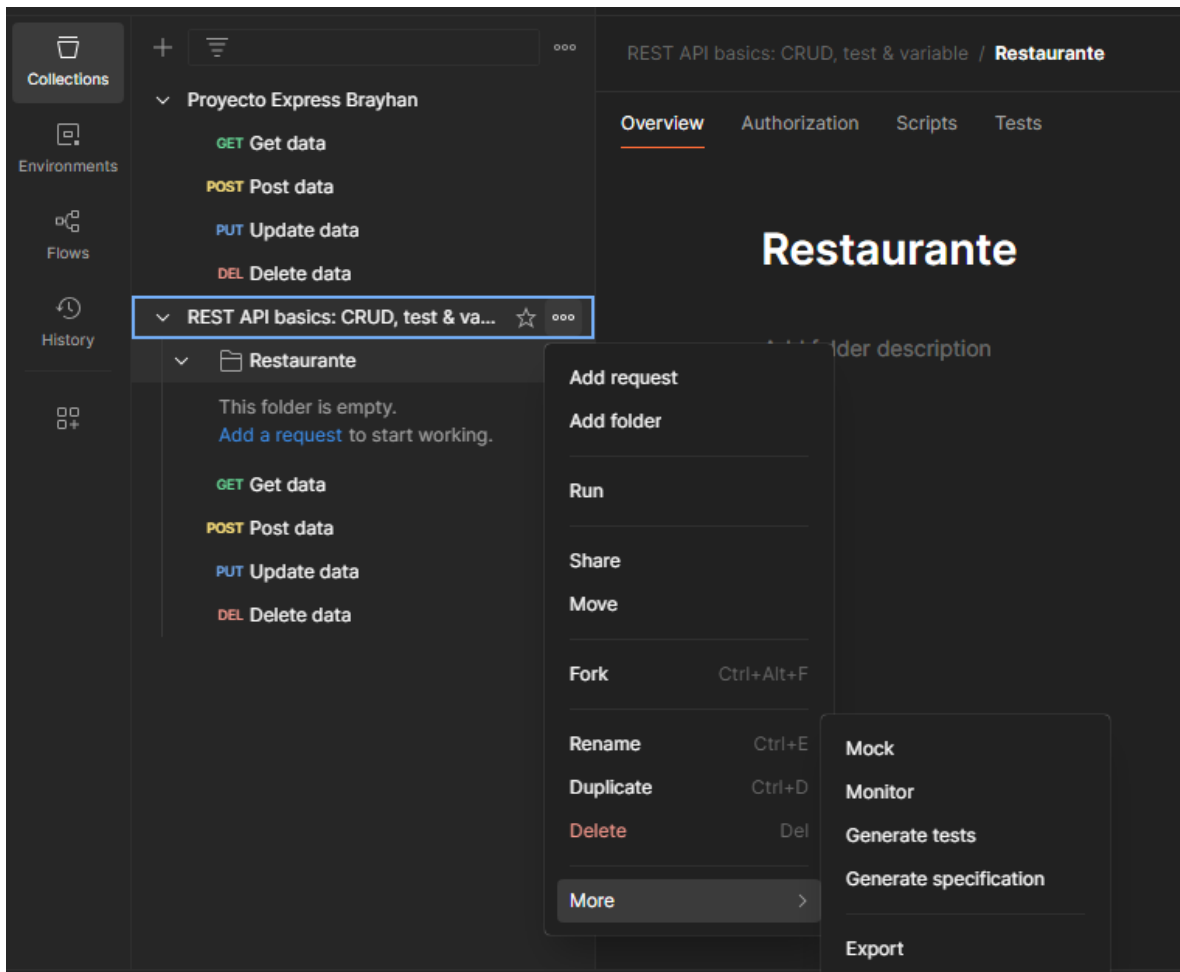
(1, 'La Parrilla de Juan', 'Buenos Aires', 'Av. Rivadavia 1234', '2015-03-21'),
(2, 'Sabor Peruano', 'Lima', 'Calle Los Pinos 450', '2018-07-10'),
(3, 'El Buen Gusto', 'Bogotá', 'Cra 10 #15-34', '2020-01-15'),
(4, 'Pizza Planet', 'Rosario', 'San Martín 789', '2019-09-12'),
(5, 'Don Mario', 'Santiago', 'Av. Providencia 200', '2016-06-30'),
(6, 'La Casa del Taco', 'Ciudad de México', 'Insurgentes Sur 3000', '2022-04-05'),
(7, 'Pasta y Punto', 'Montevideo', '18 de Julio 1456', '2014-11-09'),
(8, 'Grill & Beer', 'Buenos Aires', 'Córdoba 556', '2013-02-12'),
(9, 'Sushi Express', 'Valparaíso', 'Pedro Montt 890', '2020-08-20'),
(10, 'Asado Criollo', 'Córdoba', 'Colón 200', '2017-03-18'),
(11, 'Burgers & More', 'Medellín', 'Calle 50 #45-10', '2018-12-03'),
(12, 'Sabores del Mar', 'Cartagena', 'Bocagrande 300', '2021-01-20'),
(13, 'Veggie Life', 'Quito', 'Av. Amazonas N25', '2019-05-01'),

Conexión a base de datos PostgreSQL (Supabase).

```
$ dbjs > ...  
1 import postgres from 'postgres'; You, anteayer * agregando todo ... "postgres": Unknown word.  
2 const sql = postgres('postgres://postgres.tufetgaistnsfbdvbe:EFwjA9NwH1p6ruXw@aws-0-sa-east-1.pooler.supabase.com:5432/postgres');  
3 export default sql;  
4  
5 //postgres://postgres.tufetgaistnsfbdvbe:EFwjA9NwH1p6ruXw@aws-0-sa-east-1.pooler.supabase.com:5432/postgres "postgresql":
```

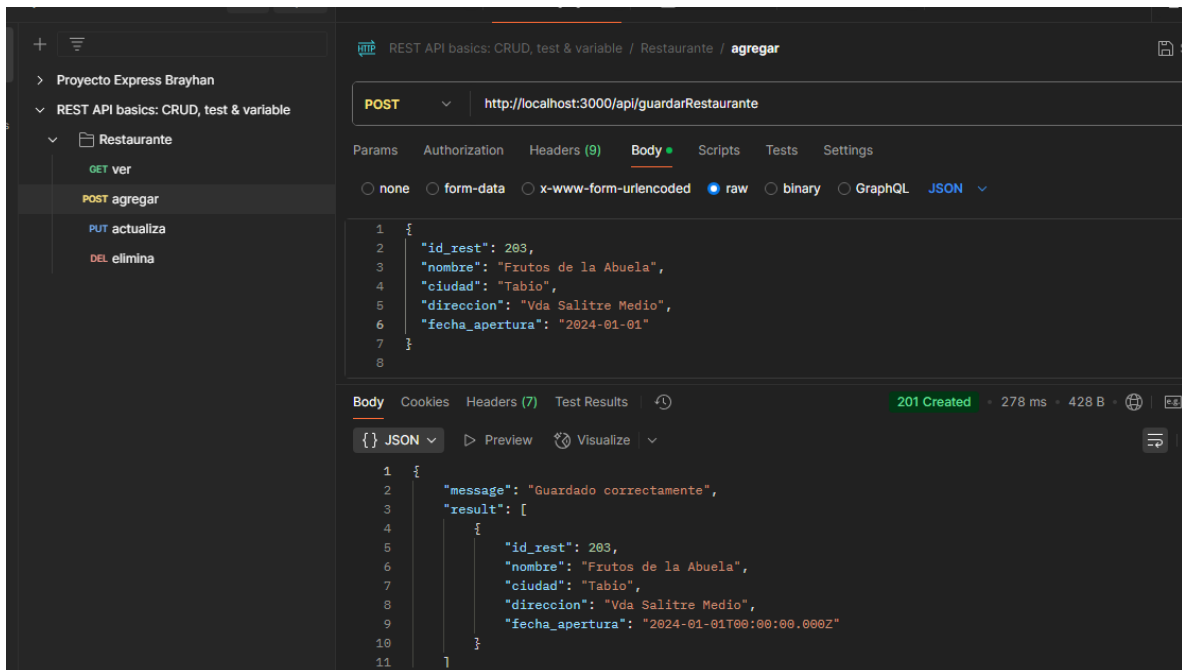
Crear rutas y controladores con Express para cada entidad (CRUD).

Restaurante

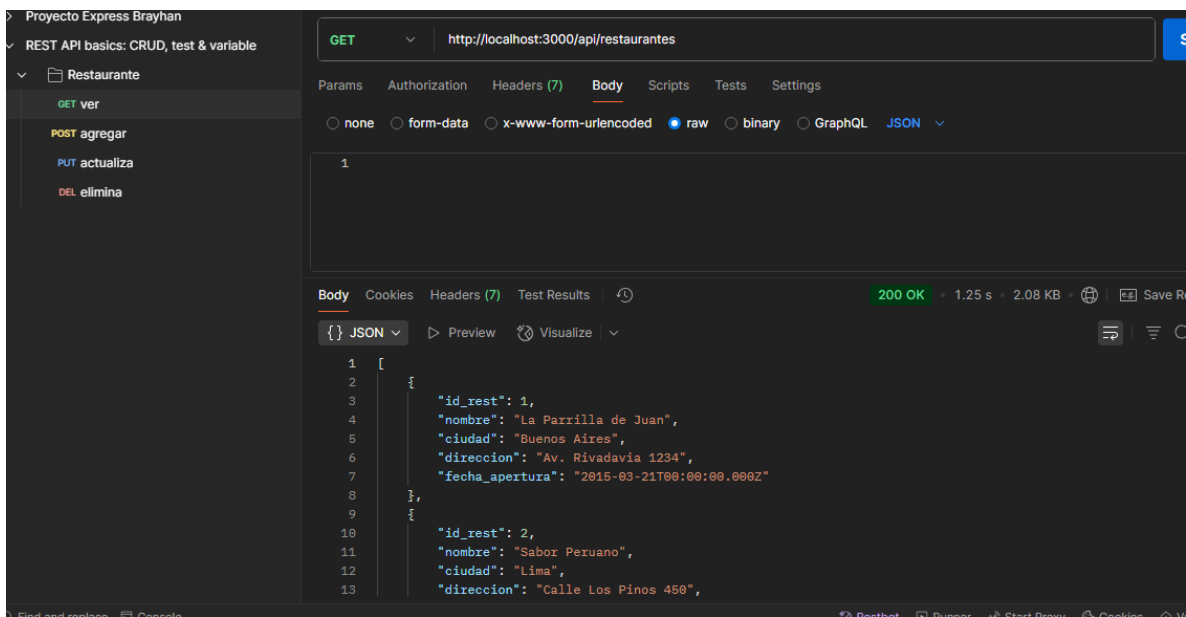


Agregar Restaurantes

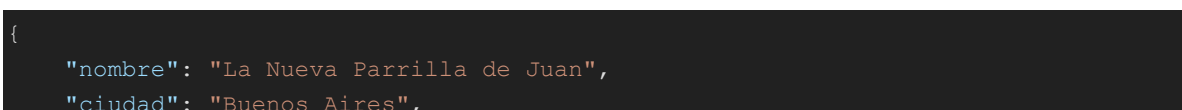
```
{
  "id_rest": 203,
  "nombre": "Frutos de la Abuela",
  "ciudad": "Tabio",
  "direccion": "Vda Salitre Medio",
  "fecha_apertura": "2024-01-01"
}
```



Ver los Restaurantes



Actualizar Restaurantes



```
"direccion": "Av. Rivadavia 1235",
"fecha_apertura": "2016-04-22"
}
```

Proyecto Express Brayhan

REST API basics: CRUD, test & variable

- Restaurante
 - GET ver
 - POST agregar
 - PUT actualiza
 - DEL elimina

PUT `http://localhost:3000/api/restaurantes/1`

Params Authorization Headers (9) Body Scripts Tests Settings

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

```
1 {
2   "nombre": "La Nueva Parrilla de Juan",
3   "ciudad": "Buenos Aires",
4   "direccion": "Av. Rivadavia 1235",
5   "fecha_apertura": "2016-04-22"
6 }
```

Body Cookies Headers (7) Test Results 200 OK · 1.16 s · 453 B Save Res

{ } JSON Preview Visualize

```
1 {
2   "message": "Restaurante actualizado correctamente",
3   "restaurante": {
4     "id_rest": 1,
5     "nombre": "La Nueva Parrilla de Juan",
6     "ciudad": "Buenos Aires",
7     "direccion": "Av. Rivadavia 1235",
8     "fecha_apertura": "2016-04-22T00:00:00.000Z"
9   }
10 }
```

Eliminar restaurantes

Proyecto Express Brayhan

REST API basics: CRUD, test & variable

- Restaurante
 - GET ver
 - POST agregar
 - PUT actualiza
 - DEL elimina

DELETE `http://localhost:3000/api/restaurantes/3`

Params Authorization Headers (7) Body Scripts Tests Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results 200 OK · 261 ms · 408 B Save Res

{ } JSON Preview Visualize

```
1 {
2   "message": "Restaurante eliminado",
3   "data": {
4     "id_rest": 3,
5     "nombre": "El Buen Gusto",
6     "ciudad": "Bogotá",
7     "direccion": "Cra 10 #15-34",
8     "fecha_apertura": "2020-01-15T00:00:00.000Z"
9   }
10 }
```

Producto

Crear productos

Proyecto Express Brayhan

REST API basics: CRUD, test & variable

- Restaurante
- Producto
 - GET VER
 - POST aGREGAR**
 - PUT New Request
 - DEL New Request

POST http://localhost:3000/api/productos

Params Authorization Headers (9) **Body** Scripts Settings

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

```
1 {
2   "id_prod": 101,
3   "nombre": "Helado de Fresa",
4   "precio": 4.50
5 }
6
```

Body Cookies Headers (7) Test Results **201 Created** · 1.69 s · 350 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Producto creado correctamente",
3   "data": {
4     "id_prod": 101,
5     "nombre": "Helado de Fresa",
6     "precio": "4.50"
7   }
8 }
```

Actualizar producto

REST API basics: CRUD, test & variable

- Restaurante
- Producto
 - GET VER
 - POST aGREGAR
 - PUT Actualiza**
 - DEL New Request

PUT http://localhost:3000/api/productos/101

Params Authorization Headers (9) **Body** Scripts Settings

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

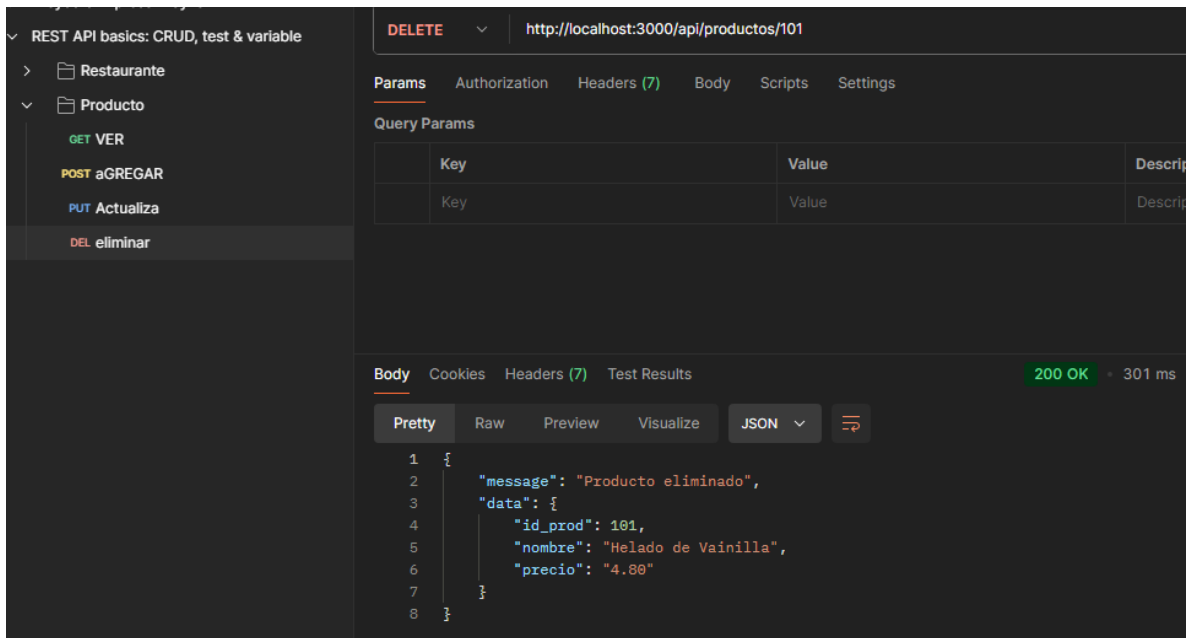
```
1 {
2   "nombre": "Helado de Vainilla",
3   "precio": 4.80
4 }
5
```

Body Cookies Headers (7) Test Results **200 OK** ·

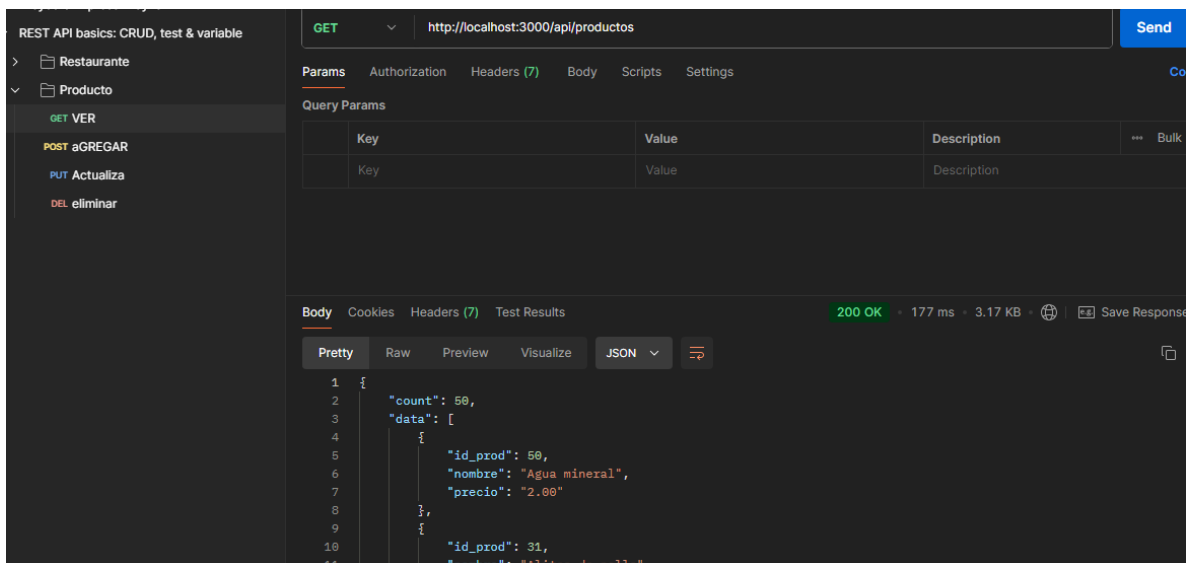
Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Producto actualizado",
3   "data": {
4     "id_prod": 101,
5     "nombre": "Helado de Vainilla",
6     "precio": "4.80"
7   }
8 }
```

Eliminar producto



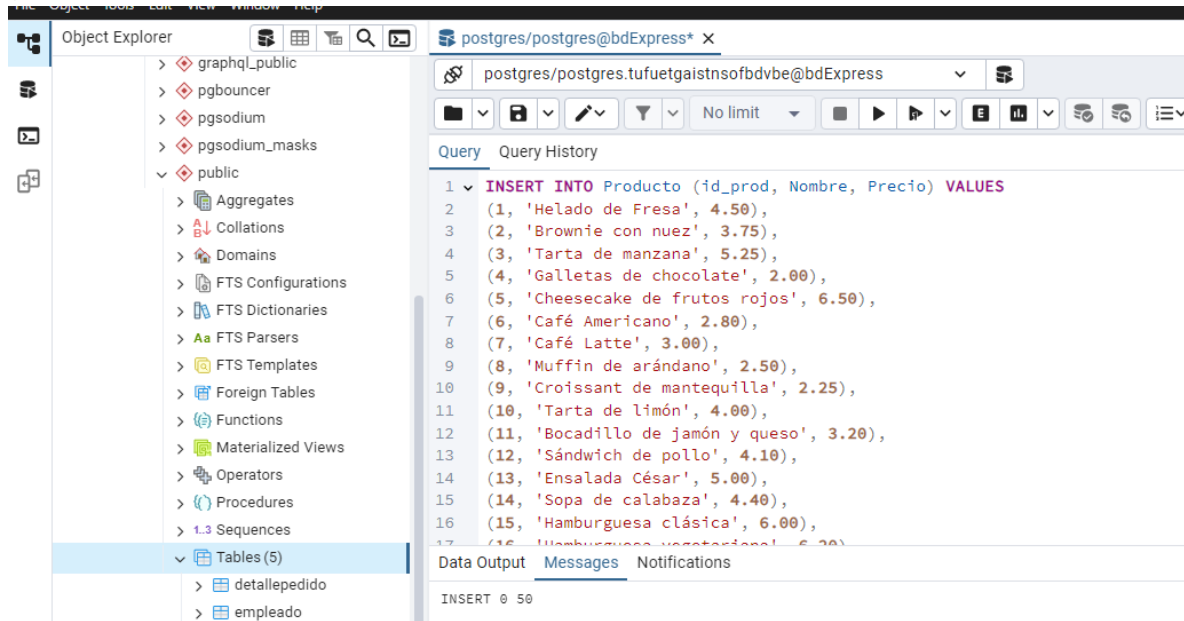
Mostrar



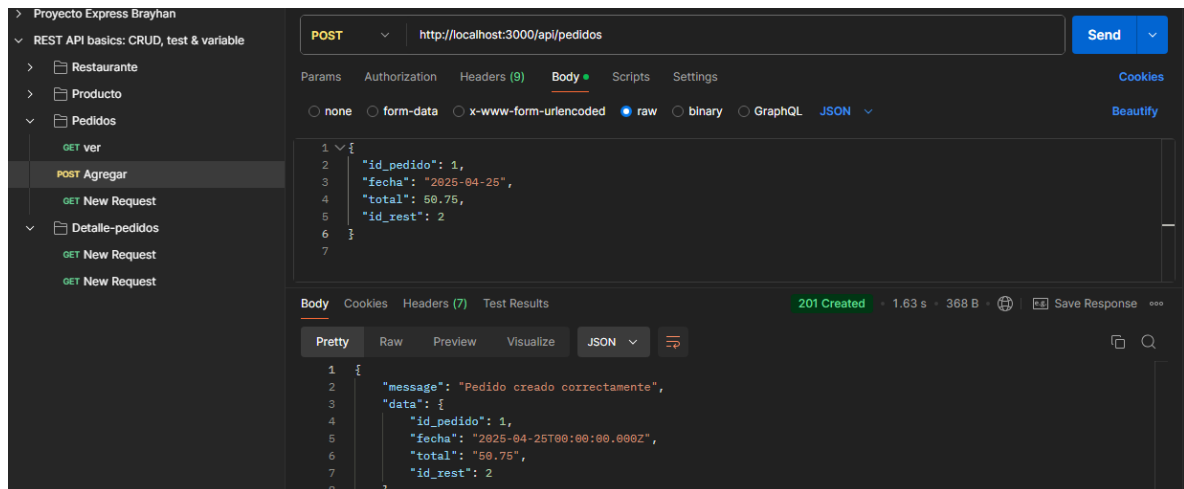
50 Productos

INSERT INTO Producto (id_prod, Nombre, Precio) VALUES

(1, 'Helado de Fresa', 4.50);



Pedido
agregar



Actualizar pedido

The screenshot shows the REST Client interface with a PUT request to `http://localhost:3000/api/pedidos/50`. The request body is a JSON object:

```
{  "id_prod": 2,  "cantidad": 3,  "subtotal": 60.00}
```

The response is a 200 OK status with a response time of 2.50 s and a size of 370 B. The response body is a JSON object:

```
{  "message": "Pedido actualizado correctamente",  "data": {    "id_pedido": 50,    "fecha": "2025-04-25T00:00:00.000Z",    "total": "150.00",    "id_rest": 1  }}
```

Eliminar pedido

The screenshot shows the REST Client interface with a DELETE request to `http://localhost:3000/api/pedidos/50`. The response is a 200 OK status with a response time of 2.50 s and a size of 370 B. The response body is a JSON object:

```
{  "message": "Pedido eliminado correctamente",  "data": {    "id_pedido": 50,    "fecha": "2025-04-25T00:00:00.000Z",    "total": "150.00",    "id_rest": 1  }}
```

Registros en pedidos

pgsodium_masks

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (5)

detallepedido

empleado

pedido

producto

restaurante

Query Query History

1 SELECT * FROM pedido;

Data Output Messages Notifications

Showing rows: 1 to 51 Page No: 1 of 1

	id_pedido [PK] integer	fecha date	total numeric (10,2)	id_rest integer
1	1	2025-04-25	50.75	2
2	50	2024-01-01	120.50	1
3	51	2024-01-02	95.75	2
4	52	2024-01-03	89.99	3
5	53	2024-01-04	150.00	4
6	54	2024-01-05	105.30	5
7	55	2024-01-06	77.80	6
8	56	2024-01-07	98.00	7
9	57	2024-01-08	66.60	8
10	58	2024-01-09	180.00	9
11	59	2024-01-10	110.10	10
12	60	2024-01-11	130.90	11

Detalles de pedido

INSERT INTO DetallePedido (id_detalle, cantidad, subtotal, id_pedido, id_prod) VALUES

(3, 3, 45.00, 51, 2);

pgsodium_masks

public

Aggregates

Collations

Domains

FTS Configurations

FTS Dictionaries

FTS Parsers

FTS Templates

Foreign Tables

Functions

Materialized Views

Operators

Procedures

Sequences

Tables (5)

detallepedido

empleado

pedido

producto

Query Query History

1 INSERT INTO DetallePedido (id_detalle, cantidad, subtotal, id_pedido, id_prod) VALUES

(3, 3, 45.00, 51, 2),

(4, 2, 38.00, 52, 7),

(5, 1, 12.99, 53, 4),

(6, 4, 52.00, 53, 9),

(7, 2, 18.00, 54, 10),

(8, 1, 20.00, 55, 12),

(9, 3, 33.00, 56, 14),

(10, 2, 25.00, 56, 3),

(11, 1, 14.00, 57, 6),

(12, 5, 60.00, 57, 11),

(13, 2, 16.00, 58, 8),

(14, 3, 33.00, 59, 13),

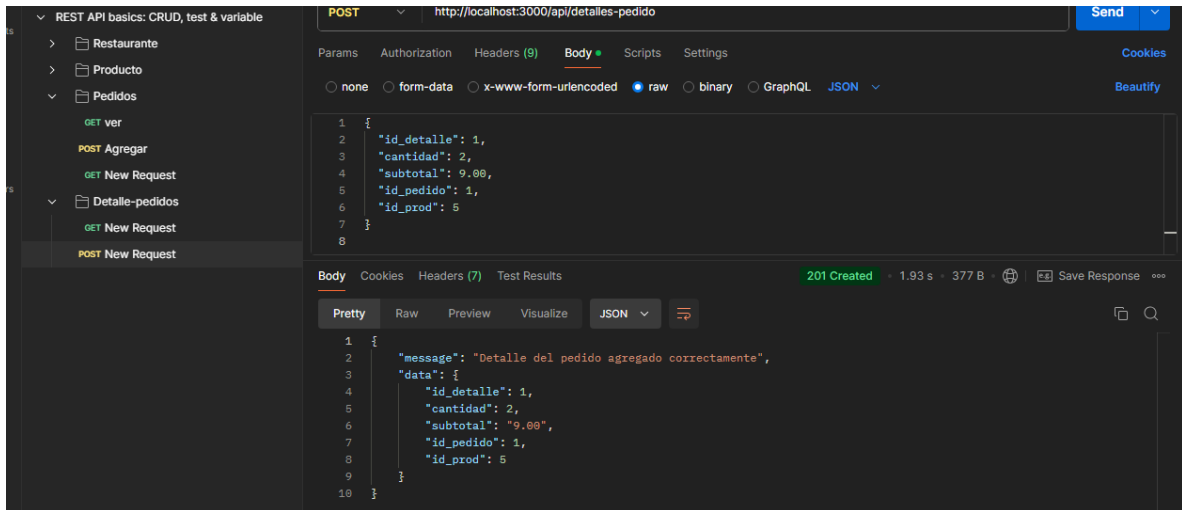
(15, 1, 10.00, 60, 16),

Data Output Messages Notifications

INSERT 0 48

Query returned successfully in 252 msec.

Agregar detalles



POST <http://localhost:3000/api/detalles-pedido> Send

Params Authorization Headers (9) Body Scripts Settings Cookies Beautify

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

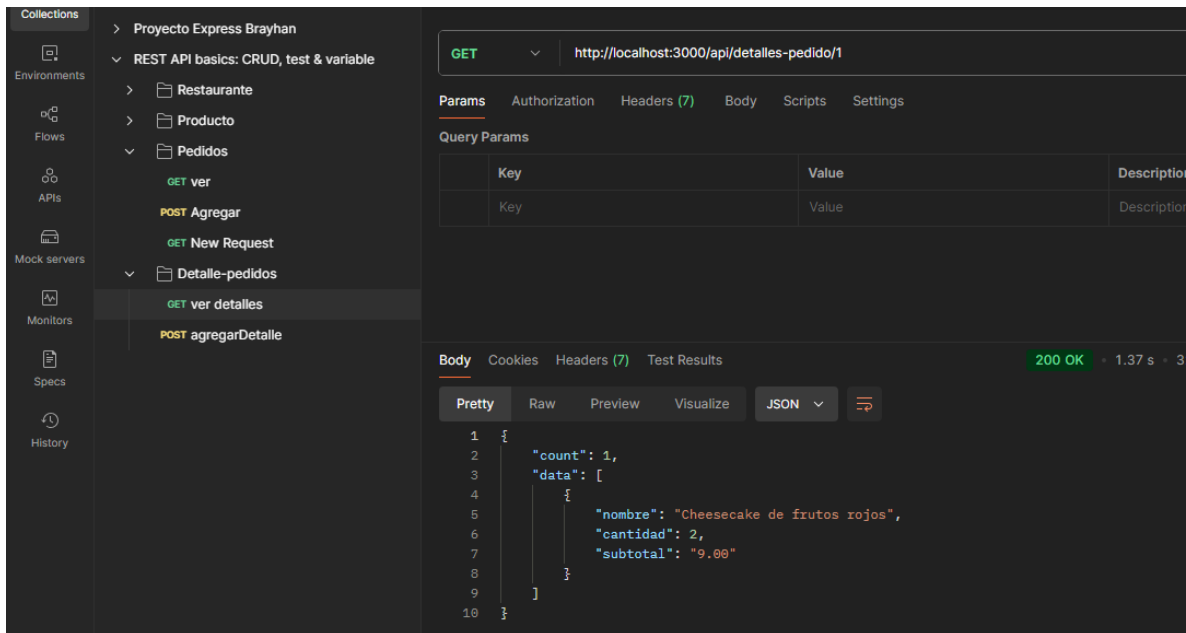
```
1 {
2   "id_detalle": 1,
3   "cantidad": 2,
4   "subtotal": 9.00,
5   "id_pedido": 1,
6   "id_prod": 5
7 }
8
```

Body Cookies Headers (7) Test Results 201 Created · 1.93 s · 377 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "message": "Detalle del pedido agregado correctamente",
3   "data": {
4     "id_detalle": 1,
5     "cantidad": 2,
6     "subtotal": "9.00",
7     "id_pedido": 1,
8     "id_prod": 5
9   }
10 }
```

Ver detalles:



GET <http://localhost:3000/api/detalles-pedido/1>

Params Authorization Headers (7) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results 200 OK · 1.37 s · 377 B

Pretty Raw Preview Visualize JSON

```
1 {
2   "count": 1,
3   "data": [
4     {
5       "nombre": "Cheesecake de frutos rojos",
6       "cantidad": 2,
7       "subtotal": "9.00"
8     }
9   ]
10 }
```

Actualizar detalles

The screenshot shows a REST client interface with a sidebar on the left containing a project tree. The main area displays a PUT request to `http://localhost:3000/api/detalles-pedido/5`. The request body is a JSON object: `{ "cantidad": 3, "subtotal": 45.00 }`. The response is a 200 OK status with a response time of 1.74 s and a body size of 363 B. The response body is shown in JSON format: `{ "message": "Detalle del pedido actualizado", "data": { "id_detalle": 5, "cantidad": 3, "subtotal": "45.00", "id_pedido": 53, "id_prod": 4 } }`.

```
PUT http://localhost:3000/api/detalles-pedido/5

{
  "cantidad": 3,
  "subtotal": 45.00
}
```

200 OK • 1.74 s • 363 B

```
{
  "message": "Detalle del pedido actualizado",
  "data": {
    "id_detalle": 5,
    "cantidad": 3,
    "subtotal": "45.00",
    "id_pedido": 53,
    "id_prod": 4
  }
}
```

Eliminar detalles:

The screenshot shows a REST client interface with a sidebar on the left containing a project tree. The main area displays a DELETE request to `http://localhost:3000/api/detalles-pedido/3`. The response is a 200 OK status with a response time of 1.68 s and a body size of 361 B. The response body is shown in JSON format: `{ "message": "Detalle del pedido eliminado", "data": { "id_detalle": 3, "cantidad": 3, "subtotal": "45.00", "id_pedido": 51, "id_prod": 2 } }`.

```
DELETE http://localhost:3000/api/detalles-pedido/3
```

200 OK • 1.68 s • 361 B

```
{
  "message": "Detalle del pedido eliminado",
  "data": {
    "id_detalle": 3,
    "cantidad": 3,
    "subtotal": "45.00",
    "id_pedido": 51,
    "id_prod": 2
  }
}
```

Empleado

The screenshot shows a database management interface. On the left, a tree view displays the database structure, including a 'public' schema with various objects like Aggregates, Collations, Domains, FTS Configurations, FTS Dictionaries, FTS Parsers, FTS Templates, Foreign Tables, Functions, Materialized Views, Operators, Procedures, Sequences, and Tables (5). The 'Tables (5)' folder is expanded, showing 'detallepedido', 'empleado', 'pedido', 'producto', and 'restaurante'. The main pane displays a SQL query in the 'Query' tab:

```
1 INSERT INTO Empleado (id_empleado, Nombre, Rol, id_rest) VALUES
2 (1, 'Juan Pérez', 'Cocinero', 1),
3 (2, 'Ana López', 'Mesera', 1),
4 (3, 'Carlos Gómez', 'Cocinero', 2),
5 (4, 'Marta Sánchez', 'Mesera', 2),
6 (5, 'José Rodríguez', 'Administrador', 3),
7 (6, 'María García', 'Cocinero', 1),
8 (7, 'Luis Fernández', 'Mesero', 1),
9 (8, 'Pablo López', 'Cocinero', 3),
10 (9, 'Laura Martín', 'Mesera', 2),
11 (10, 'Raúl Pérez', 'Administrador', 1),
```

The 'Query History' tab shows the same query. The 'Data Output' tab shows the result: 'INSERT 0 50'. The 'Messages' tab shows the message: 'Query returned successfully in 381 msec.'

Agregar

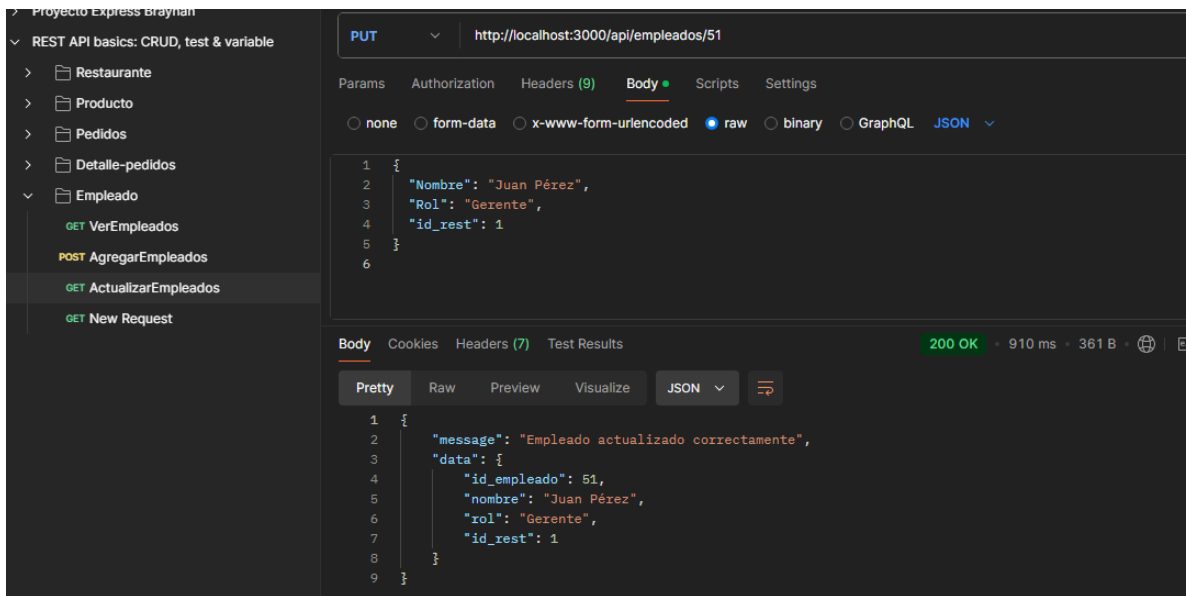
The screenshot shows a REST client interface. The left sidebar displays a project structure for 'Proyecto Express Brayhan' with a 'REST API basics: CRUD, test & variable' section. The 'Empleado' endpoint is selected, showing a list of methods: 'GET VerEmpleados', 'GET AgregarEmpleados', 'GET ActualizarEmpleados', and 'GET New Request'. The main pane shows a 'POST' request to 'http://localhost:3000/api/empleados'. The 'Body' tab is active, showing a JSON payload:

```
1 {
2   "id_empleado": 51,
3   "Nombre": "Juan Rodriguez",
4   "Rol": "Cocinero",
5   "id_rest": 1
6 }
```

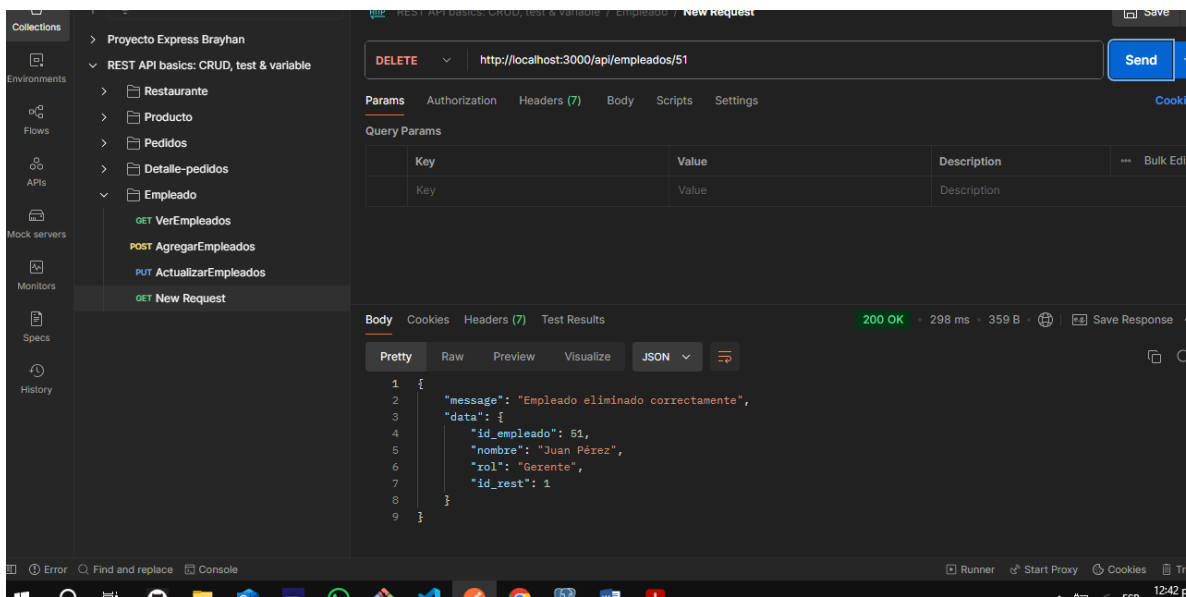
The 'Body' tab also shows the response: '201 Created' with a status of '1.59 s' and a size of '366 B'. The 'Pretty' tab is active, showing the response JSON:

```
1 {
2   "message": "Empleado creado correctamente",
3   "data": {
4     "id_empleado": 51,
5     "nombre": "Juan Rodriguez",
6     "rol": "Cocinero",
7     "id_rest": 1
8   }
9 }
```

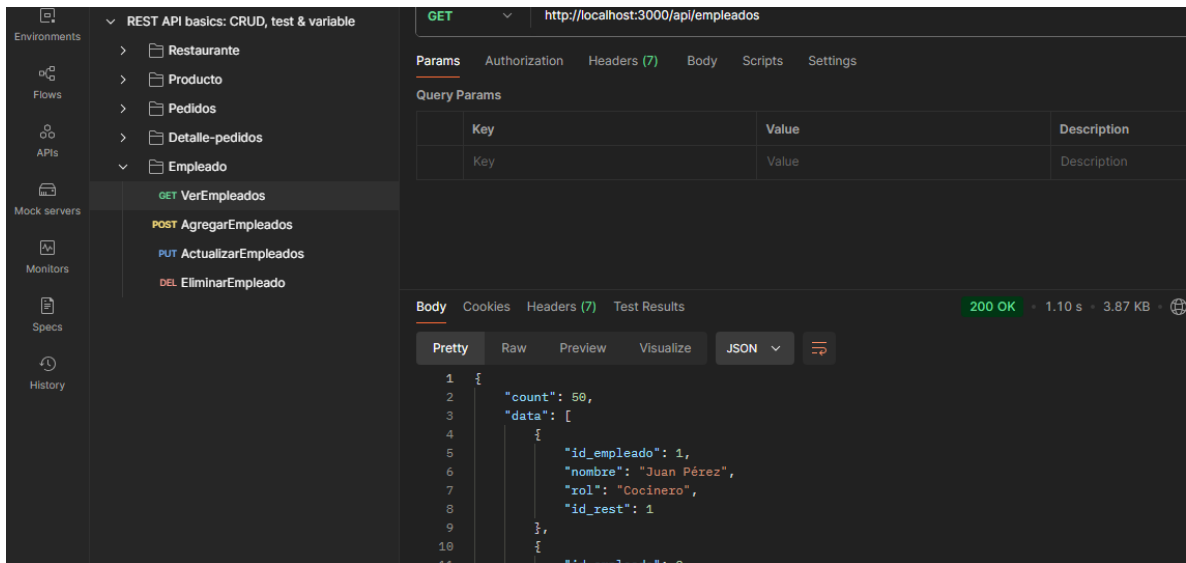
Actualizar



Eliminar

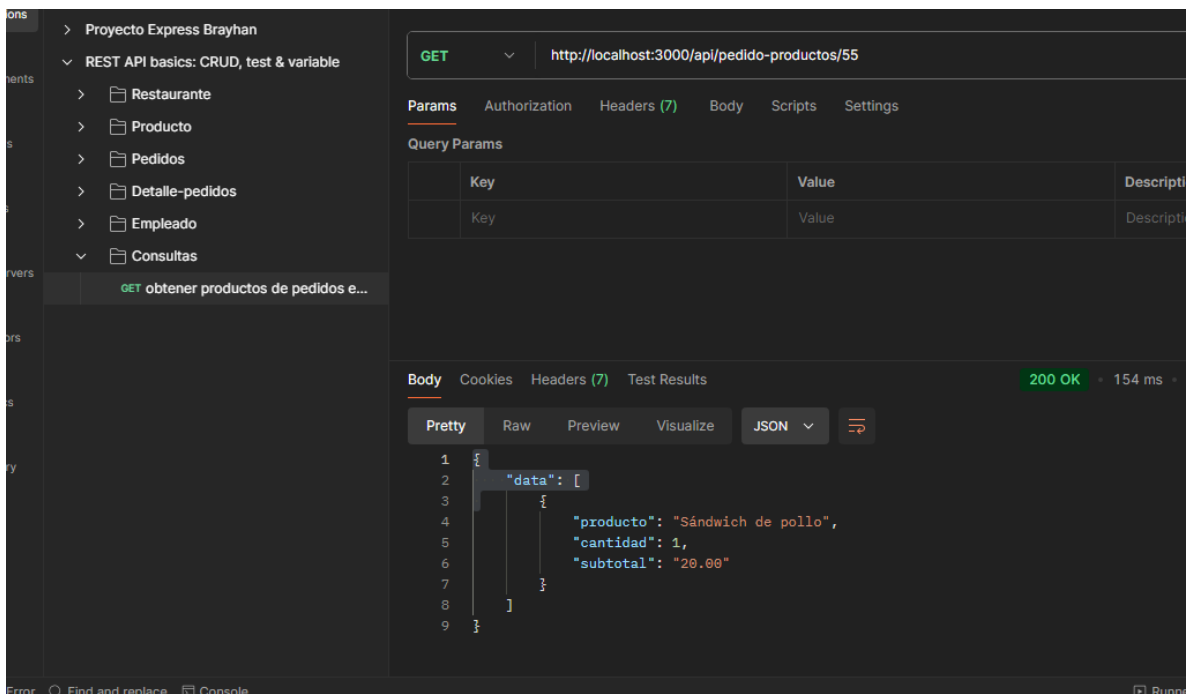


Obtener



Implementar las siguientes consultas nativas y exponerlas por rutas:

- 1) obtener productos de pedidos especificos



2) Obtener los productos más vendidos (más de X unidades)

REST Client interface showing a GET request to `http://localhost:3000/api/productos-mas-vendidos?unidades=2`. The response is 200 OK, 163 ms, 1007 B. The response body is a JSON array of products with their counts.

```
1 {
2   "count": 13,
3   "data": [
4     {
5       "producto": "Galletas de chocolate",
6       "unidades_vendidas": "3"
7     },
8     {
9       "producto": "Sushi Maki",
10      "unidades_vendidas": "3"
11    }
12  ]
13 }
```

3) Obtener el total de ventas por restaurante

REST Client interface showing a GET request to `http://localhost:3000/api/ventas-por-restaurante`. The response is 200 OK, 1.04 s, 2.67 KB. The response body is a JSON array of restaurants with their total sales.

```
1 {
2   "count": 43,
3   "data": [
4     {
5       "restaurante": "Sabor Peruano",
6       "total_ventas": "343.65"
7     },
8     {
9       "restaurante": "Pizza Planet",
10      "total_ventas": "280.70"
11    }
12  ]
13 }
```

4) Obtener los pedidos realizados en una fecha específica

REST API basics: CRUD, test & variable / Consultas / **Obtener los pedidos realizados en una fecha específica**

GET

Params Authorization Headers (7) Body Scripts Settings

Query Params

Key	Value	Description
Key	Value	Description

Body Cookies Headers (7) Test Results **200 OK** · 1.54 s · 336 B ·

Pretty Raw Preview Visualize JSON

```
1 {
2   "count": 1,
3   "data": [
4     {
5       "id_pedido": 1,
6       "fecha": "2025-04-25T00:00:00.000Z",
7       "total": "150.00",
8       "id_rest": 2
9     }
10  ]
11 }
```

5) Obtener los empleados por rol en un restaurante

REST API basics: CRUD, test & variable / Consultas / **Obtener los empleados por rol en un restaurante**

GET **Send**

Params Authorization Headers (7) Body Scripts Settings **Cookies**

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (7) Test Results **200 OK** · 1.84 s · 423 B · Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": [
3     {
4       "rol": "Mesera",
5       "cantidad_empleados": "3"
6     },
7     {
8       "rol": "Mesero",
9       "cantidad_empleados": "3"
10    }
11  ]
12 }
```