

---

# Torneo de Pokémon Go

*JDBC & Psycopg*

---



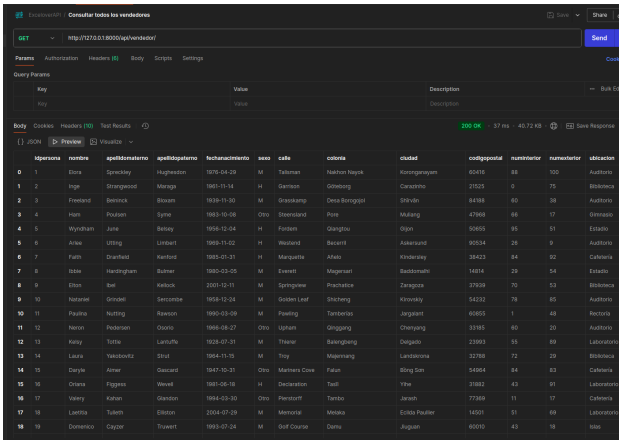
Para esta práctica decidimos usar *Psycopg2* y *Django*. En el proyecto hay instrucciones de cómo instalar y correr el proyecto.

## API'S EN POSTMAN

### Vendedor

#### GET

Probamos la API para la tabla **Vendedor** consultando a todos los alimentos con GET. Recibimos el código 200, lo que significa que la solicitud se procesó correctamente y el servidor devolvió una respuesta exitosa. En SQL haríamos `SELECT * FROM vendedor;`



idpersona	nombre	apellidomaterno	apellidopaterno	fechanacimiento	sexo	calle	colonia	ciudad	codigopostal	numinterior	numexterior	ubicacion
1	Elena	Spencer	Hughes	1975-04-28	M	Taberna	Nahua Naya	Acapulco	60410	85	100	Auditorio
2	Hugo	Strangeway	Morgan	1981-01-14	M	Garcera	Ginebra	Caracas	21025	0	75	Biblioteca
3	Frederick	Baltica	Brown	1939-11-30	M	Grasskamp	Dea-Birgipg	Shirah	84188	60	38	Auditorio
4	Hani	Poulsen	Eghe	1983-10-08	OTV	Steensland	Pore	Mulang	47988	66	17	Estadio
5	Wyndham	Jane	Reley	1998-12-04	M	Fosden	Qianghu	Dipn	50655	95	51	Estadio
6	Arkie	Villeg	Leibert	1995-11-02	M	Wolstead	Bazant	Auerland	91014	26	9	Auditorio
7	Patty	Shirah	Harlow	1981-06-28	M	Marquette	Jahy	Wendway	28421	84	82	Cafeteria
8	Walter	Warrington	Bulmer	1993-02-05	M	Fawcett	Nagasaki	Bachmann	14814	29	54	Estadio
9	Eliot	Ree	Holcomb	2005-10-11	M	Springme	Pranostine	Zhangou	37038	70	53	Biblioteca
10	Nahomi	Gonzalez	Servicio	1938-12-14	M	Golden Leaf	Shiweing	Kinoway	54232	78	60	Auditorio
11	Pauline	Nutting	Reason	1990-03-08	M	Peeling	Tanburles	Jergahen	60805	1	48	Restaurante
12	Noron	Podman	Quinn	1988-08-27	OTV	Upham	Qinggang	Chengyang	31165	60	20	Auditorio
13	Kenny	Torres	Luttrell	1928-07-31	M	Truett	Baerghang	Delgado	21093	95	88	Laboratorio
14	Laura	Yakoboff	Stal	1984-11-16	M	Tray	Marmang	Leidennia	32788	72	29	Biblioteca
15	Barth	Amel	Chenard	1947-06-20	OTV	Martha Cove	Falen	Wing-Sun	34984	84	83	Cafeteria
16	Charles	Pegues	Wood	1991-06-18	M	Thompson	Todd	Vita	71881	43	90	Laboratorio
17	Wesley	Kahan	Chenard	1984-10-30	OTV	Pierceoff	Tanbur	Jarrah	77369	11	17	Cafeteria
18	Laurita	Talbot	Edman	2004-07-29	M	Merritt	Mexica	Edna-Plester	14001	91	68	Laboratorio
19	Domenica	Cayor	Truett	1993-07-24	M	Golf Course	Datto	Jugan	60010	43	18	Restaurante

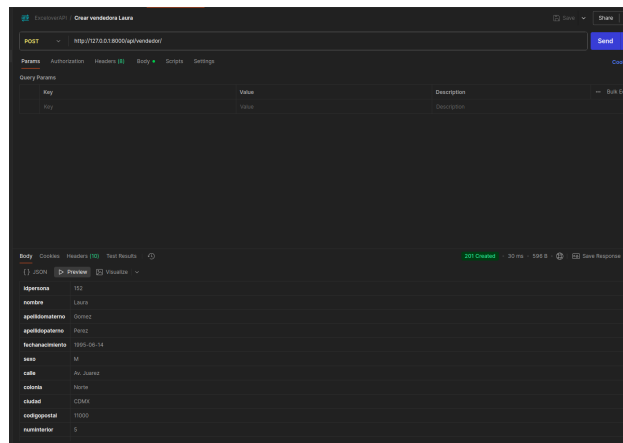
**Figura 1:** Operación GET para obtener a todos los vendedores

#### POST

Recibimos el código 201, lo que significa que el servidor creó de forma exitosa un nuevo recurso.

En SQL:

```
INSERT INTO Vendedor (
    idpersona, nombre, apellidomaterno, apellidopaterno, fechanacimiento,
    sexo, calle, colonia, ciudad, codigopostal, numinterior, numexterior, ubicacion
) VALUES (
    151, 'Laura', 'Gomez', 'Perez', '1995-06-14',
    'M', 'Av. Juarez', 'Norte', 'CDMX', 11000, 5, 33, 'Cafeteria'
);
```

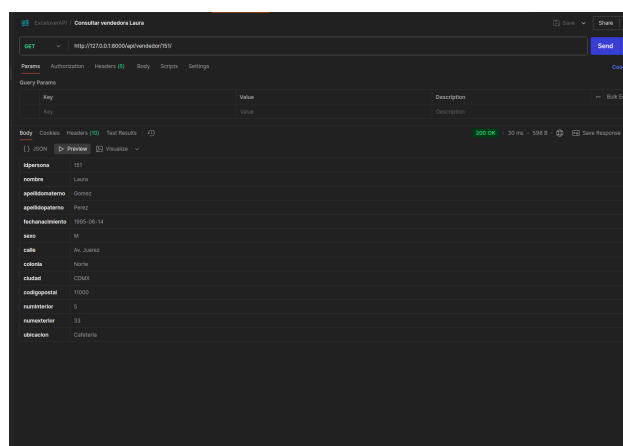


**Figura 2:** Operación POST para crear a la vendedora Laura

## GET

Recibimos el código 200, lo que significa que la solicitud se procesó correctamente y el servidor devolvió una respuesta exitosa.

En SQL: `SELECT * FROM Vendedor WHERE idpersona = 151;`

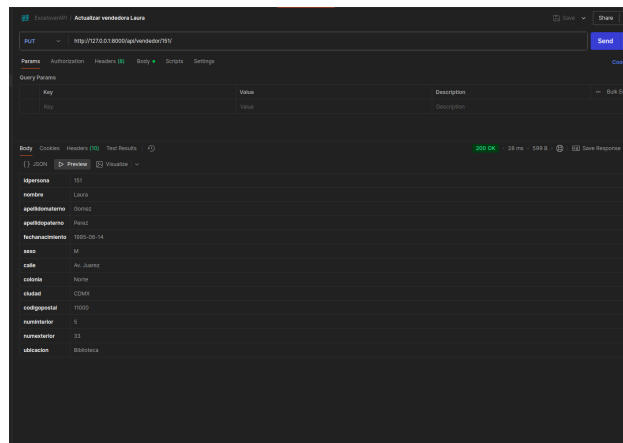


**Figura 3:** Operación GET para obtener a la vendedora Laura

## PUT

Recibimos el código 200, lo que significa que la solicitud se procesó correctamente y el servidor devolvió una respuesta exitosa.

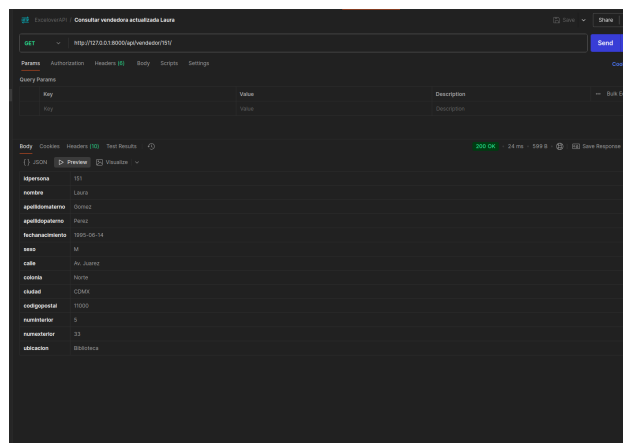
En SQL: `UPDATE Vendedor SET ubicacion = 'Biblioteca' WHERE idpersona = 151;`



**Figura 4:** Operación PUT para actualizar a la vendedora Laura

**GET** Recibimos el código 200, lo que significa que la solicitud se procesó correctamente y el servidor devolvió una respuesta exitosa.

En SQL: `SELECT * FROM Vendedor WHERE idpersona = 151;`

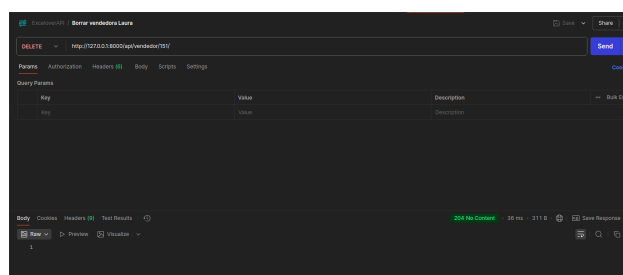


**Figura 5:** Operación GET para consultar a la vendedora Laura tras actualización

## DELETE

Recibimos el código 204, lo que significa que el servidor borró de forma exitosa.

En SQL: `DELETE FROM Vendedor WHERE idpersona = 151;`

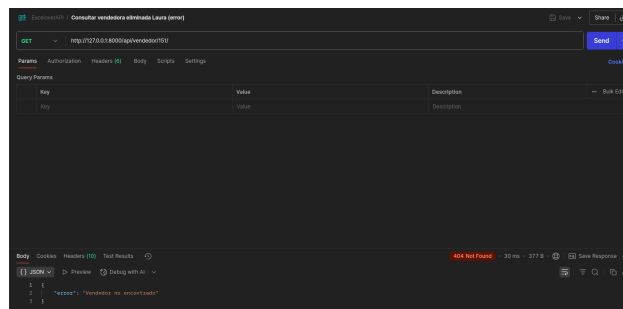


**Figura 6:** Operación DELETE para borrar a la vendedora Laura

## GET

Recibimos el código 404, lo que significa que el servidor no encontró el recurso, tal como esperábamos.

En SQL: `SELECT * FROM Vendedor WHERE idpersona = 151;`



**Figura 7:** Operación GET para consultar a la vendedora Laura después de borrarse

## Alimento

### GET

Recibimos el código 200, lo que significa que la solicitud se procesó correctamente y el servidor devolvió una respuesta exitosa.

En SQL: `SELECT * FROM Alimento;`

The screenshot shows a REST client interface with a GET request to the URL `http://192.168.1.8000/alimento`. The status bar at the bottom indicates a 200 OK response. The response body is a JSON array of food items, which is displayed as a table in the interface.

id	fecha de caducidad	nombre	tipo	precio	id persona
0	2028-12-20	Margarita rosa	Dulce	975	122
1	2027-02-25	Helado de vainilla	Pastete	222.06	13
2	2028-01-11	Galletas de avena	Embutido	376.94	145
3	2025-11-09	Café molido	Sopa	89.96	109
4	2028-06-21	Tomate	Vendado	308.73	56
5	2028-04-26	Pasta espagueti	Grano	148.76	37
6	2027-01-02	Sandwich	Pasta	476.21	41
7	2027-10-18	Queso de barra	Pasta	491.3	81
8	2028-01-09	Alfalfa	Ensalada	191.89	101
9	2027-01-15	Tarta de pastel	Panaderia	147.9	134
10	2027-09-28	Helado de vainilla	Dulce	61.09	85
11	2027-03-30	Sopa instantanea	Candados	154.29	98
12	2027-01-31	Salsa de tomate	Panaderia	230.92	101
13	2028-08-14	Ensalada	Lacteos	276.84	26
14	2027-08-05	Yogurt natural	Cereal	433.11	3
15	2027-09-24	Mermelada	Pasta	138.34	142
16	2025-11-02	Sopa instantanea	Candados	144.14	89
17	2028-01-09	Almuerzo	Vendado	386.77	143
18	2028-12-24	Copa de helado	Asado	107.19	16

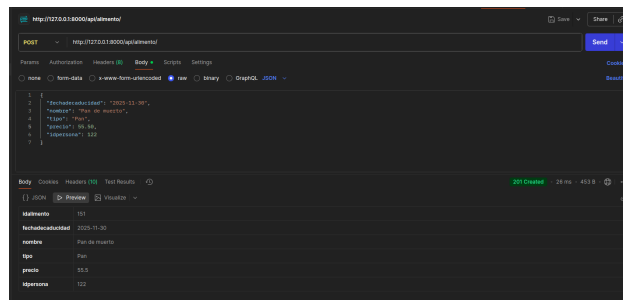
**Figura 8:** Operación GET para obtener todos los alimentos

### POST

Recibimos el código 201, lo que significa que el servidor creó de forma exitosa un nuevo recurso.

En SQL:

```
INSERT INTO Alimento (
    idalimento, fechadecaducidad, nombre, tipo, precio, idpersona
) VALUES (
    151, '2025-11-30', 'Pan de muerto', 'Pan', 55.5, 122
);
```

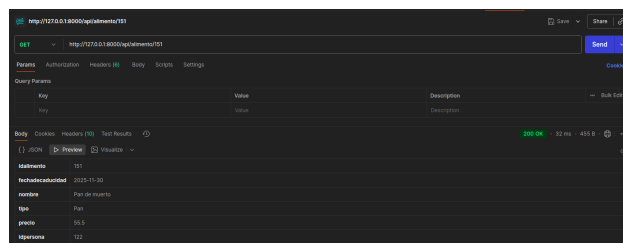


**Figura 9:** Operación POST para crear el alimento Pan de muerto

## GET

Recibimos el código 200, lo que significa que la solicitud se procesó correctamente y el servidor devolvió una respuesta exitosa.

En SQL: `SELECT * FROM Alimento WHERE idalimento = 151;`

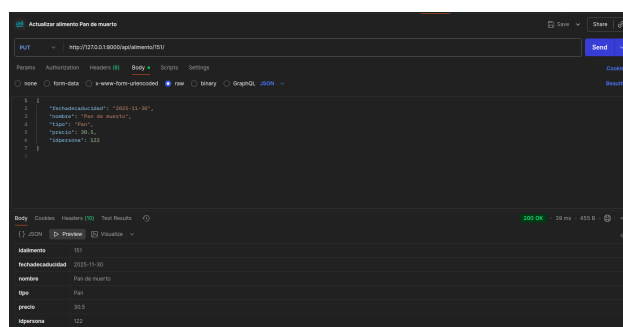


**Figura 10:** Operación GET para obtener el alimento Pan de muerto

## PUT

Recibimos el código 200, lo que significa que la solicitud se procesó correctamente y el servidor devolvió una respuesta exitosa.

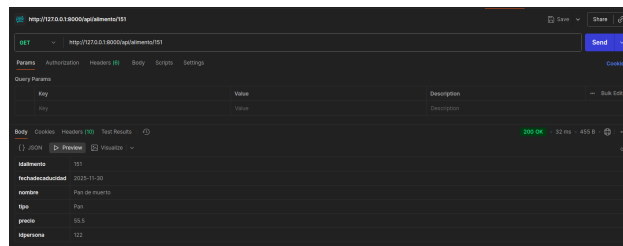
En SQL: `UPDATE Alimento SET precio = '30.5' WHERE idalimento = 151;`



**Figura 11:** Operación PUT para actualizar el alimento Pan de muerto

## GET

En SQL: `SELECT * FROM Alimento WHERE idalimento = 151;`

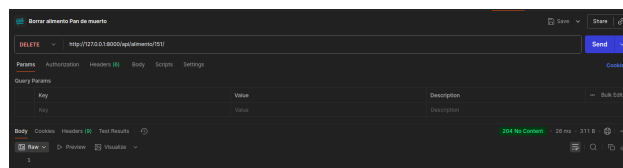


**Figura 12:** Operación GET para obtener el alimento Pan de muerto

## DELETE

Recibimos el código 204, lo que significa que el servidor borró de forma exitosa.

En SQL: `DELETE FROM Alimento WHERE idalimento = 151;`

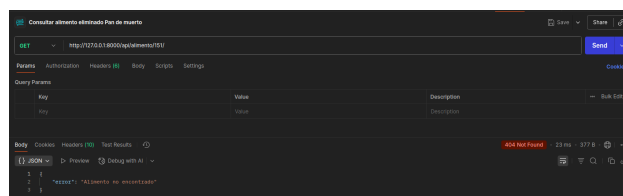


**Figura 13:** Operación DELETE para borrar al alimento Pan de muerto

## GET

Recibimos el código 404, lo que significa que el servidor no encontró el recurso, tal como esperábamos.

En SQL: `SELECT * FROM Alimento WHERE idalimento = 151;`



**Figura 14:** Operación GET para consultar el alimento eliminado Pan de muerto