

Uso de endpoints en postman:

Base: http://0.0.0.0:5000

Nota: todos los endpoints contemplan un .lower() para que el usuario “Vanesa” sea igual a “vanesa” y a “vANesA”, para evitar posibles duplicados por error de tipeo humano.

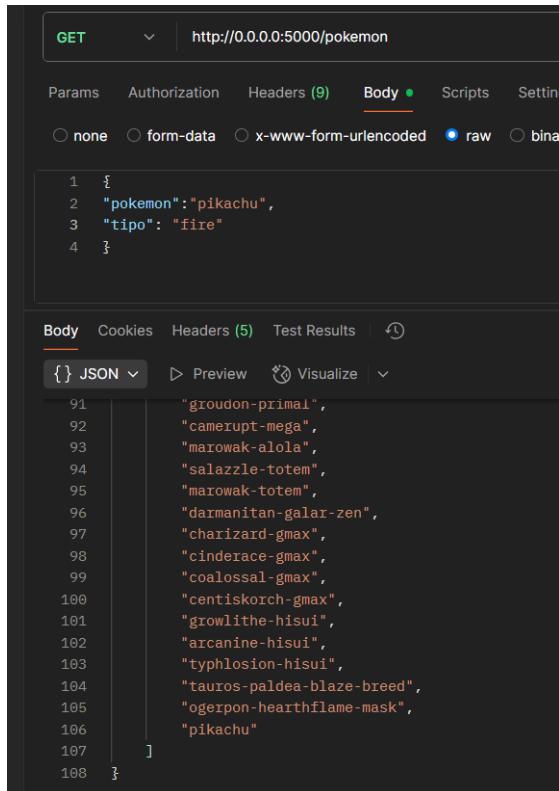
GET:

✿ /pokemon: devuelve una lista con el pokemon ingresado, todos los Pokemones del tipo ingresado o ambas unidas.

⇒ body: {"pokemon": "nombrePokemon", "tipo": "tipoPokemones"} (opcional al menos uno)

Casos de prueba:

-Ingreso de nombre de pokemon y tipo (no el mismo tipo del pokemon ingresado), en este caso son todos los pokemones tipo fuego + pikachu



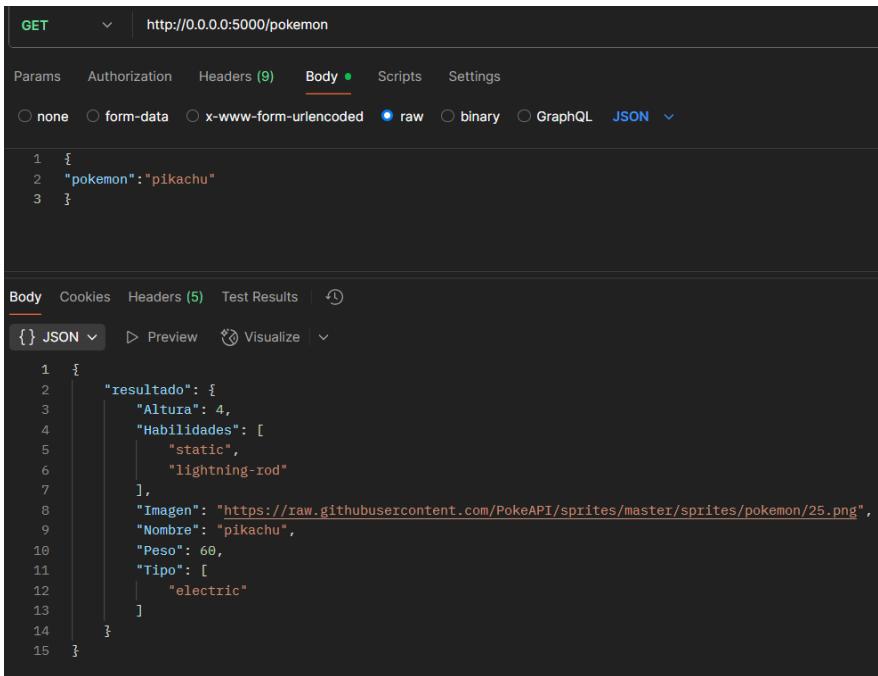
```
1  {
2   "pokemon": "pikachu",
3   "tipo": "fire"
4 }
```

Body Cookies Headers (5) Test Results ⏱

{ } JSON ▾ ▷ Preview ⚡ Visualize ▾

```
91  "groudon-primal",
92  "camerupt-mega",
93  "marowak-alola",
94  "salazzle-totem",
95  "marowak-totem",
96  "darmanitan-galar-zen",
97  "charizard-gmax",
98  "cinderace-gmax",
99  "coalossal-gmax",
100  "centiskorch-gmax",
101  "growlithe-hisui",
102  "arcanine-hisui",
103  "typhlosion-hisui",
104  "tauros-paldea-blaze-breed",
105  "ogerpon-hearthflame-mask",
106  "pikachu"
107  ]
108 }
```

- Ingreso solo de nombre del Pokémon: (funciona de igual forma si dejamos tipo vacío)



GET | http://0.0.0.0:5000/pokemon

Params Authorization Headers (9) Body Scripts Settings

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

```

1  {
2   "pokemon": "pikachu"
3  }

```

Body Cookies Headers (5) Test Results | ⏱

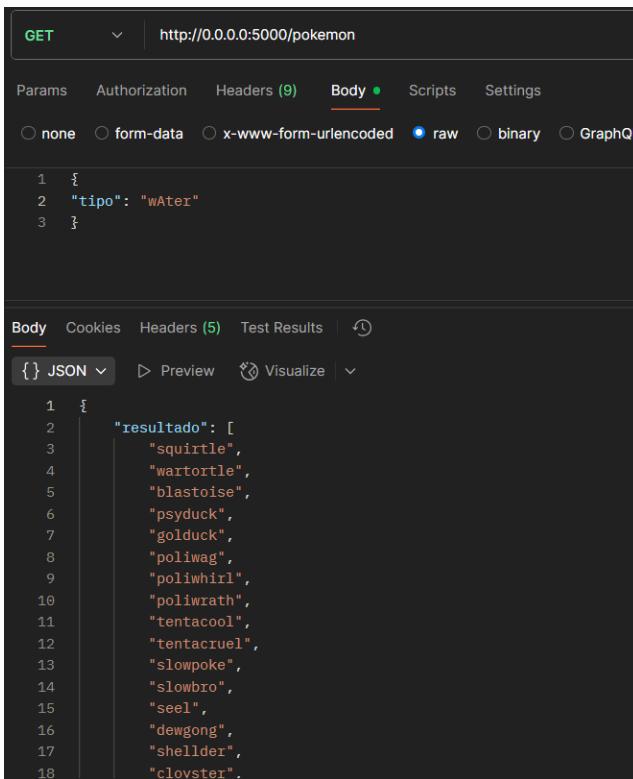
{ } JSON ▾ ▷ Preview ⚡ Visualize ▾

```

1  {
2   "resultado": {
3     "Altura": 4,
4     "Habilidades": [
5       "static",
6       "lightning-rod"
7     ],
8     "Imagen": "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/25.png",
9     "Nombre": "pikachu",
10    "Peso": 60,
11    "Tipo": [
12      "electric"
13    ]
14  }
15 }

```

- Ingreso de tipo: (funciona también si dejamos el nombre vacío)



GET | http://0.0.0.0:5000/pokemon

Params Authorization Headers (9) Body Scripts Settings

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

```

1  {
2   "tipo": "wAtEr"
3  }

```

Body Cookies Headers (5) Test Results | ⏱

{ } JSON ▾ ▷ Preview ⚡ Visualize ▾

```

1  {
2   "resultado": [
3     "squirtle",
4     "wartortle",
5     "blastoise",
6     "psyduck",
7     "golduck",
8     "poliwag",
9     "poliwhirl",
10    "poliwrath",
11    "tentacool",
12    "tentacruel",
13    "slowpoke",
14    "slowbro",
15    "seel",
16    "dewgong",
17    "shellder",
18    "cloyster".

```

- No ingreso ningún dato:

GET | http://0.0.0.0:5000/pokemon

Params Authorization Headers (9) Body Scripts Settings

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

```

1  {
2    "tipo": "",
3    "pokemon": ""
4  }

```

Body Cookies Headers (5) Test Results | ⏱

{ } JSON ▾ Preview ⚡ Visualize | ▾

```

1  {
2    "error": "Se debe ingresar al menos un dato"
3  }

```

✿ /favoritos : devuelve la lista de favoritos del usuario ingresado, con nombre de los pokemones e id de cada uno.

⇒ body: {"usuario":"nombre"}

Casos de prueba:

- Ingreso usuario con pokemones en favoritos:

GET | http://0.0.0.0:5000/favoritos

Params Authorization Headers (9) Body Scripts Settings

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

```

1  {
2    "usuario": "VanEsa"
3  }

```

Body Cookies Headers (5) Test Results | ⏱

{ } JSON ▾ Preview ⚡ Visualize | ▾

```

1  {
2    "favoritos": [
3      {
4        "id": 1,
5        "nombre": "charmander"
6      }
7    ]
8  }

```

- Ingreso usuario sin pokemones favoritos:

```

GET http://0.0.0.0:5000/favoritos
Body
Params Authorization Headers (9) Body Scripts Settings
none form-data x-www-form-urlencoded raw bin
1 {
2   "usuario": "Vanesa"
3 }

Body Cookies Headers (5) Test Results
[] JSON Preview Visualize
1 {
2   "error": "No tenés lista de favoritos"
3 }

```

- No ingreso del dato:

```

GET http://0.0.0.0:5000/favoritos
Body
Params Authorization Headers (9) Body Scripts Settings
none form-data x-www-form-urlencoded raw bin
1 {
2   "usuario": ""
3 }

Body Cookies Headers (5) Test Results
[] JSON Preview Visualize
1 {
2   "error": "Tenés que ingresar el usuario"
3 }

```

✿ /favoritos/id : devuelve la información del Pokémón.

⇒ body: {"usuario":"nombre","id":"idDelPokemonFavorito"}

Casos de prueba:

- Ingreso correcto de los datos:

The screenshot shows a Postman interface for a GET request to `http://0.0.0.0:5000/favoritos/id`. The Body tab is selected, showing raw JSON input:

```
1 {
2   "usuario": "vanesa",
3   "id": "1"
4 }
```

The Response tab shows the JSON output:

```
1 {
2   "resultado": {
3     "id": 1,
4     "nombre": "charmander"
5   }
6 }
```

- Ingreso erróneo del id:

The screenshot shows a Postman interface for a GET request to `http://0.0.0.0:5000/favoritos/id`. The Body tab is selected, showing raw JSON input:

```
1 {
2   "usuario": "vanesa",
3   "id": "3"
4 }
```

The Response tab shows the JSON output:

```
1 {
2   "error": "No se encontró el Pokémon con id 3"
3 }
```

- No ingreso de algún dato:

```

GET http://0.0.0.0:5000/favoritos/id

Params Authorization Headers (9) Body Scripts Settings
 none  form-data  x-www-form-urlencoded  raw 
1 {
2   "usuario": "vanesa",
3   "id": ""
4 }

Body Cookies Headers (5) Test Results
[] JSON Preview Visualize
1 {
2   "error": "Faltan datos"
3 }

```

✿ /cancion : te asigna un pokemon y una canción del dia utilizando el horoscopo.

⇒ body: {"usuario": "nombre", "fecha": "yyyy-mm-dd"}

Casos de prueba:

- Correcto ingreso de datos:

```

GET http://0.0.0.0:5000/cancion

Params Authorization Headers (9) Body Scripts Settings
 none  form-data  x-www-form-urlencoded  raw  binary  GraphQL  JSON
1 {
2   "usuario": "vanesa",
3   "fecha": "2025-07-25"
4 }

Body Cookies Headers (5) Test Results
[] JSON Preview Visualize
1 {
2   "Hola": "vanesa",
3   "Tu Pokémon del dia es": "numel",
4   "Tu canción del dia de One Direction es": "Live While We're Young"
5 }

```

- Fecha no válida:

```

GET | http://0.0.0.0:5000/cancion
Params Authorization Headers (9) Body Scripts Settings
 none  form-data  x-www-form-urlencoded  raw  binary  GraphQL
1 {
2   "usuario": "vanesa",
3   "fecha": "205-0-25"
4 }

Body Cookies Headers (5) Test Results
[{}]
JSON ▾ ▶ Preview ⚡ Visualize ▾
1 {
2   "error": "El formato de la fecha no es válido"
3 }

```

- Algun dato vacío:

```

GET | http://0.0.0.0:5000/cancion
Params Authorization Headers (9) Body Scripts
 none  form-data  x-www-form-urlencoded  raw
1 {
2   "usuario": "vanesa",
3   "fecha": ""
4 }

Body Cookies Headers (5) Test Results
[{}]
JSON ▾ ▶ Preview ⚡ Visualize ▾
1 {
2   "error": "Falta ingresar la fecha"
3 }

```

POST

✿ /favoritos : guarda un Pokemón a tu lista de favoritos.

⇒ body: ("usuario":"nombre", "pokemon":"nombreDelPokemon")

Casos de prueba:

- Agregar correctamente un Pokemón, datos bien ingresados:

```

POST | http://0.0.0.0:5000/favoritos
Params Authorization Headers (9) Body Scripts Settings
 none  form-data  x-www-form-urlencoded  raw  binary  GraphQL  JSON
1 {
2   "usuario": "Vanesa",
3   "pokemon": "charmander"
4 }

Body Cookies Headers (5) Test Results
[{}]
JSON ▾ ▶ Preview ⚡ Visualize ▾
1 {
2   "mensaje": "Se ha ingresado correctamente el pokemon charmander a la lista de favoritos de vanesa"
3 }

```

- Agregar algún dato mal ingresado:

```

POST http://0.0.0.0:5000/favoritos
Body
1 {
2   "usuario": "Vanesa",
3   "pokemon": "hola"
4 }

Body
1 {
2   "error": "El nombre del pokemon no existe"
3 }

```

- Agregar Pokemón ya ingresado, evitar duplicados:

```

POST http://0.0.0.0:5000/favoritos
Body
1 {
2   "usuario": "VanEsa",
3   "pokemon": "charmander"
4 }

Body
1 {
2   "mensaje": "El pokemon charmander ya estaba en la lista de favoritos de vanesa"
3 }

```

❖ /horoscopo : calcula el Pokemón que sos según tu día de nacimiento y tu signo zodiacal.

⇒ body: ("usuario": "nombre","fecha":"yyyy-mm-dd")

Casos de prueba:

- Datos correctamente ingresados:

The screenshot shows a Postman interface with a POST request to `http://0.0.0.0:5000/horoscopo`. The Body tab is selected, showing raw JSON input:

```

1  {
2    "usuario": "kira",
3    "fecha": "2022-06-02"
4  }

```

The Response tab displays the received JSON data:

```

1  {
2    "pokemon": {
3      "Altura": 3,
4      "Habilidades": [
5        "keen-eye",
6        "tangled-feet",
7        "big-pecks"
8      ],
9      "Imagen": "https://raw.githubusercontent.com/PokeAPI/sprites/master/sprites/pokemon/16.png",
10     "Nombre": "pidgey",
11     "Peso": 18,
12     "Tipo": [
13       "normal",
14       "flying"
15     ]
16   },
17   "signo": "Géminis",
18   "usuario": "kira"

```

- Falta de algún dato:

The screenshot shows a Postman interface with a POST request to `http://0.0.0.0:5000/horoscopo`. The Body tab is selected, showing raw JSON input:

```

1  {
2    "usuario": "",
3    "fecha": "2022-06-02"
4  }

```

The Response tab displays an error message:

```

1  {
2    "error": "Falta ingresar el nombre"
3  }

```

- Fecha con mal formato:

POST http://0.0.0.0:5000/horoscopo

Params Authorization Headers (9) Body Scripts Settings

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

```

1  {
2    "usuario": "kira",
3    "fecha": "2022-00-02"
4  }

```

Body Cookies Headers (5) Test Results

{ } JSON ▾ ▶ Preview ⚡ Visualize

```

1  {
2    "error": "El formato de la fecha no es válido"
3  }

```

DELETE:

✿ /favoritos : elimina el Pokemón de tu lista de favoritos.

⇒ body: {"usuario": "nombre", "id": "idDelPokemonFavorito"}

Casos de prueba:

- Eliminado efectivo:

DELETE http://0.0.0.0:5000/favoritos

Params Authorization Headers (9) Body Scripts

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

```

1  {
2    "usuario": "Vanesa",
3    "id": "1"
4  }

```

Body Cookies Headers (5) Test Results

{ } JSON ▾ ▶ Preview ⚡ Visualize

```

1  {
2    "eliminado": true
3  }

```

- Tratar de eliminar Pokémon que no existe:

The screenshot shows the Postman interface with a DELETE request to `http://0.0.0.0:5000/favoritos`. The Body tab is selected, showing raw JSON data:

```
1 {  
2   "usuario": "Vanesa",  
3   "id": "4"  
4 }
```

Below the request, the response is shown under the Body tab, also in raw JSON format:

```
{ } JSON ▾
```

```
1 {  
2   "eliminado": false  
3 }
```

- Falta de algún dato:

The screenshot shows the Postman interface with a DELETE request to `http://0.0.0.0:5000/favoritos`. The Body tab is selected, showing raw JSON data with missing fields:

```
1 {  
2   "usuario": "",  
3   "id": "1"  
4 }
```

Below the request, the response is shown under the Body tab, also in raw JSON format, indicating an error:

```
{ } JSON ▾
```

```
1 {  
2   "error": "Usuario no encontrado"  
3 }
```