

# API de Pokémon — NestJS + PostgreSQL

---

Esta API entrega un pequeño catálogo de Pokémon para el frontend del Taller 2.

La idea es replicar lo básico que se usó de PokeAPI en el Taller 1, pero sin depender de internet.

Toda la información se sirve desde una base PostgreSQL con datos precargados, incluyendo los sprites locales que se encuentran en `/img/pokemon/...`

---

## Cómo ejecutarla con Docker Compose

Desde la raíz del proyecto solo hay que ejecutar:

```
docker compose up --build
```

La API queda disponible en:

`http://localhost:3000`

Antes de iniciar, el contenedor ejecuta automáticamente:

```
npm run seed
```

para cargar los datos base de Pokémon en la BD.

---

## Endpoints

GET `/pokemon?limit=&offset=`

Entrega una lista paginada de Pokémon.

El parámetro `limit` tiene un máximo de 50.

Ejemplo de respuesta:

```
{
  "count": 30,
  "results": [
    { "name": "bulbasaur", "url": "/pokemon/1" },
    { "name": "ivysaur", "url": "/pokemon/2" }
  ]
}
```

GET `/pokemon/:idOrName`

Devuelve el detalle de un Pokémon por id numérico o por nombre (búsqueda case-insensitive).

Ejemplo de respuesta:

```
{
  "id": 25,
  "name": "pikachu",
  "height": 4,
  "weight": 60,
  "types": [
    { "slot": 1, "type": { "name": "electric" } }
  ],
  "sprites": {
    "front_default": "/img/pokemon/25.png",
    "back_default": null,
    "other": {
      "official-artwork": {
        "front_default": "/img/pokemon/25_artwork.png"
      }
    }
  },
  "stats": [
    { "base_stat": 35, "stat": { "name": "hp" } },
    { "base_stat": 55, "stat": { "name": "attack" } }
  ]
}
```

---

## Datos y base utilizada

- Base de datos: PostgreSQL
- Dato inicial: seed-data.json (se carga con npm run seed)
- Ejecución de seed:

En Docker se ejecuta automáticamente antes de levantar la API

Esquema principal: tabla pokemon con columnas:

```
id
name
sprite_default
sprite_artwork
types (jsonb)
```

```
height_m  
weight_kg  
stats (jsonb)
```

TypeORM con `synchronize=true` crea o ajusta la tabla automáticamente en desarrollo.