

Practica REST API Y CONTENEDORES.

Ulices Salvador Aguila

Contreras

1. Inicio del proyecto.

El primer paso para dar inicio al proyecto consiste en la creación del archivo "Package.json". Este archivo reflejará toda la configuración y las dependencias del proyecto y se crea ejecutando el siguiente comando:

```
$ npm init -y
```

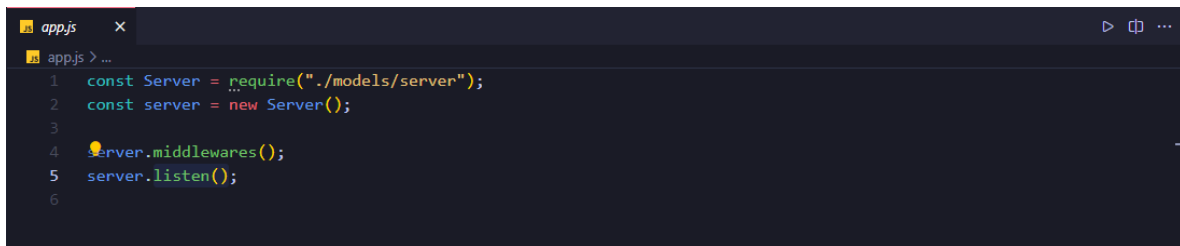
Ilustración 1. Comando para iniciar proyecto en Node.js

```
{
  "name": "sps-apirest-pt",
  "version": "1.0.0",
  "description": "",
  "main": "app.js",
  > Depurar
  "scripts": {
    "test": "echo \\\"Error: no test specified\\\" && exit 1",
    "dev": "nodemon app.js",
    "start": "node app.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "axios": "^1.5.0",
    "cors": "^2.8.5",
    "dotenv": "^16.3.1",
    "express": "^4.18.2",
    "express-validator": "^7.0.1",
    "node-fetch": "^3.3.2"
  }
}
```

Ilustración 2. Estructura de "package.json"

2. Creación del Servidor en el Archivo "App.js"

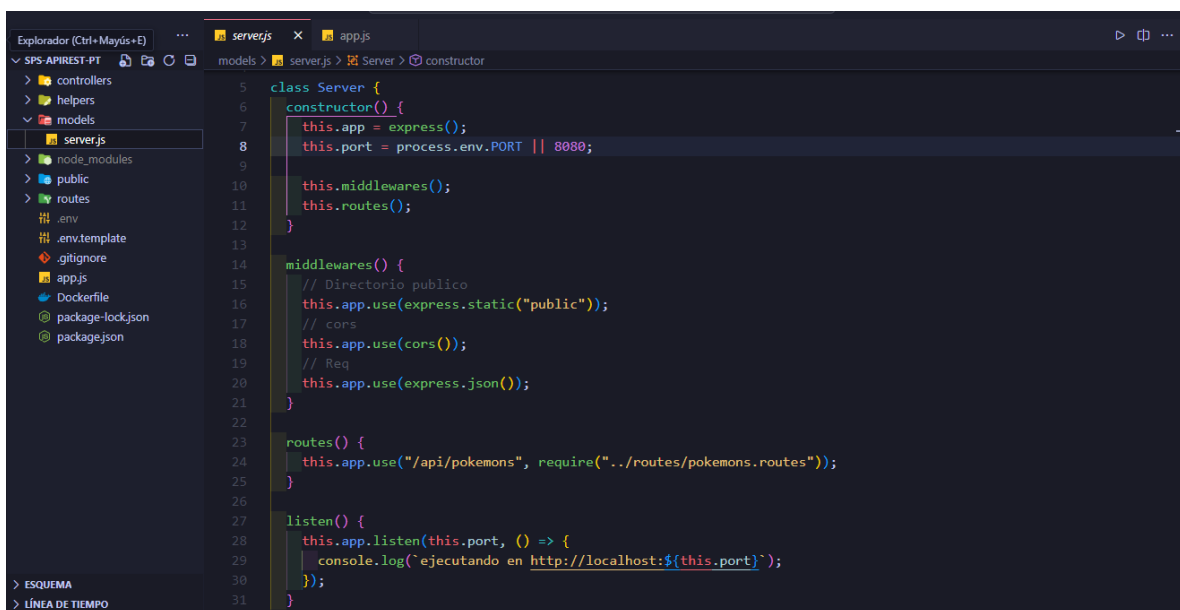
En el archivo "App.js", se define generalmente el componente principal de la aplicación y se configuran rutas, estado global y otros aspectos esenciales. En este caso, se utilizará para ejecutar funcionalidades como los middlewares y el método "listen" que inicia el servidor web y lo hace escuchar las solicitudes entrantes en un puerto específico.



```
1 const Server = require("../models/server");
2 const server = new Server();
3
4 server.middlewarees();
5 server.listen();
6
```

Ilustración 3. Archivo "App.js", archivo principal de la aplicación.

Para la funcionalidad de "App.js", se optado por crear un modelo que contendrá todas las funcionalidades necesarias para las rutas y middlewares. Este modelo se define en el archivo "server", que se crea en la carpeta "Models".



```
5 class Server {
6   constructor() {
7     this.app = express();
8     this.port = process.env.PORT || 8080;
9
10    this.middlewarees();
11    this.routes();
12  }
13
14  middlewarees() {
15    // Directorio publico
16    this.app.use(express.static("public"));
17    // cors
18    this.app.use(cors());
19    // Req
20    this.app.use(express.json());
21  }
22
23  routes() {
24    this.app.use("/api/pokemons", require("../routes/pokemons.routes"));
25  }
26
27  listen() {
28    this.app.listen(this.port, () => {
29      console.log(`ejecutando en http://localhost:${this.port}`);
30    });
31  }
32}
```

Ilustración 4. Modelo "server", configuraciones principales de la aplicación (middlewarees, rutas, listen).

3. Creación de rutas para peticiones GET.

Para crear las rutas, se realiza una carpeta llamada "Routes". En esta carpeta, se encuentra un archivo llamado "pokemons.routes.js" donde se colocarán todas las rutas necesarias para diferentes peticiones, como:

- **/:** Mostrar todos los pokemons.
- **/:name:** Mostrar pokemons por su nombre.
- **/:specie:** Mostrar pokemon por especie.



Ilustración 5. Rutas para peticiones GET.

Estas rutas se concatenan con la ruta principal:

- **/api/pokemons**

Para realizar peticiones en diferentes rutas, se debe hacer de la siguiente manera:

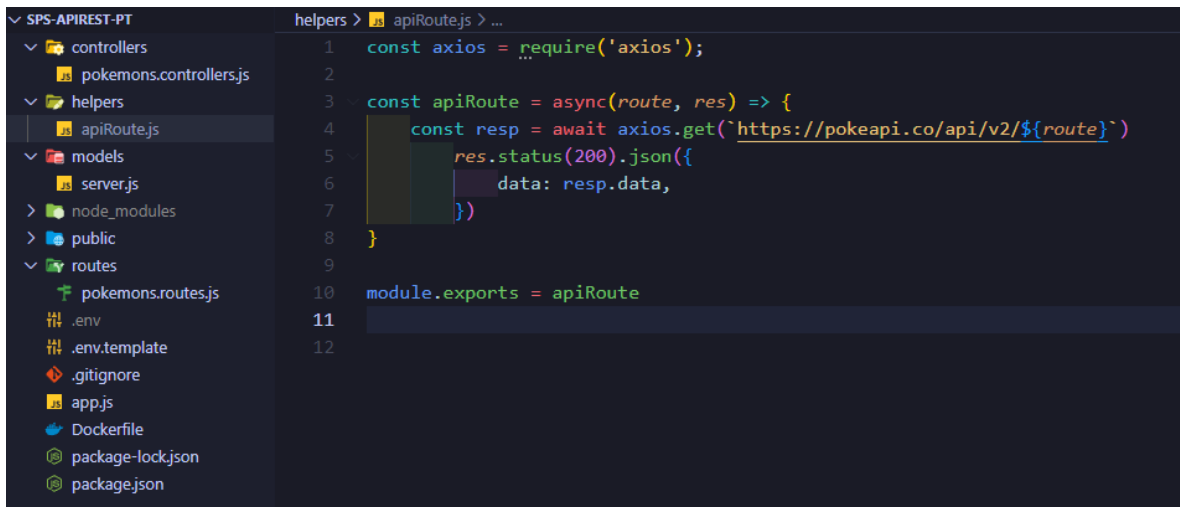
- <http://localhost:8090/api/pokemons>
- <http://localhost:8090/api/pokemons/pikachu>
- <http://localhost:8090/api/pokemons/pokemon-species/wormadam>

4. Creación de controlador de Pokémon, y consumo de api externa.

Para crear el controlador, primero se crea un script de ayuda llamado "apiRoute.js", donde se implementa la funcionalidad para realizar peticiones a APIs externas utilizando el paquete Axios. Los pasos son los siguientes:

Se especifica la API a consumir y se concatena el tipo de petición que se realizará, ejemplo:

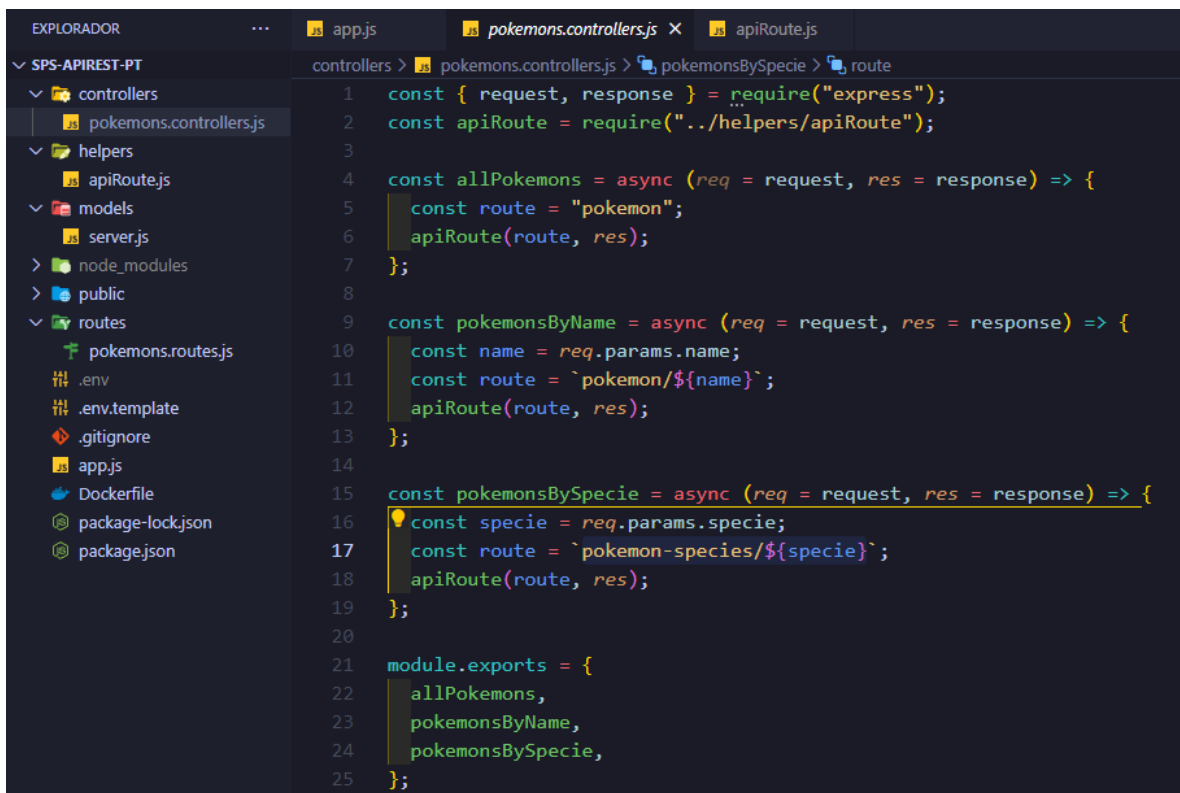
- pokemon: Todos los Pokémon
- pokemon/\${name}: Pokémon por nombre
- pokemon-species/\${specie}: Pokémon por especie



```
1 const axios = require('axios');
2
3 const apiRoute = async (route, res) => {
4   const resp = await axios.get(`https://pokeapi.co/api/v2/${route}`)
5   res.status(200).json({
6     data: resp.data,
7   })
8 }
9
10 module.exports = apiRoute
11
12
```

Ilustración 6. Configuración de axios, para realizar diferentes peticiones, dependiendo de la ruta que se le otorgue.

La respuesta de la llamada a la API ya contiene una variable llamada "data", que se devuelve al cliente en una función junto con la respuesta y el estatus exitoso de la petición. Esta función se exporta.



```
1 const { request, response } = require("express");
2 const apiRoute = require("../helpers/apiRoute");
3
4 const allPokemons = async (req = request, res = response) => {
5   const route = "pokemon";
6   apiRoute(route, res);
7 };
8
9 const pokemonsByName = async (req = request, res = response) => {
10   const name = req.params.name;
11   const route = `pokemon/${name}`;
12   apiRoute(route, res);
13 };
14
15 const pokemonsBySpecie = async (req = request, res = response) => {
16   const specie = req.params.specie;
17   const route = `pokemon-species/${specie}`;
18   apiRoute(route, res);
19 };
20
21 module.exports = {
22   allPokemons,
23   pokemonsByName,
24   pokemonsBySpecie,
25 };

```

Ilustración 7. Funciones con la "data" referente a diferentes peticiones GET.

5. Pruebas en Postman como cliente.

Una vez definidas las rutas como se muestra en el punto 3, se realizan las respectivas peticiones en Postman, lo que permite mostrar los resultados según la ruta a la que se realice la petición.

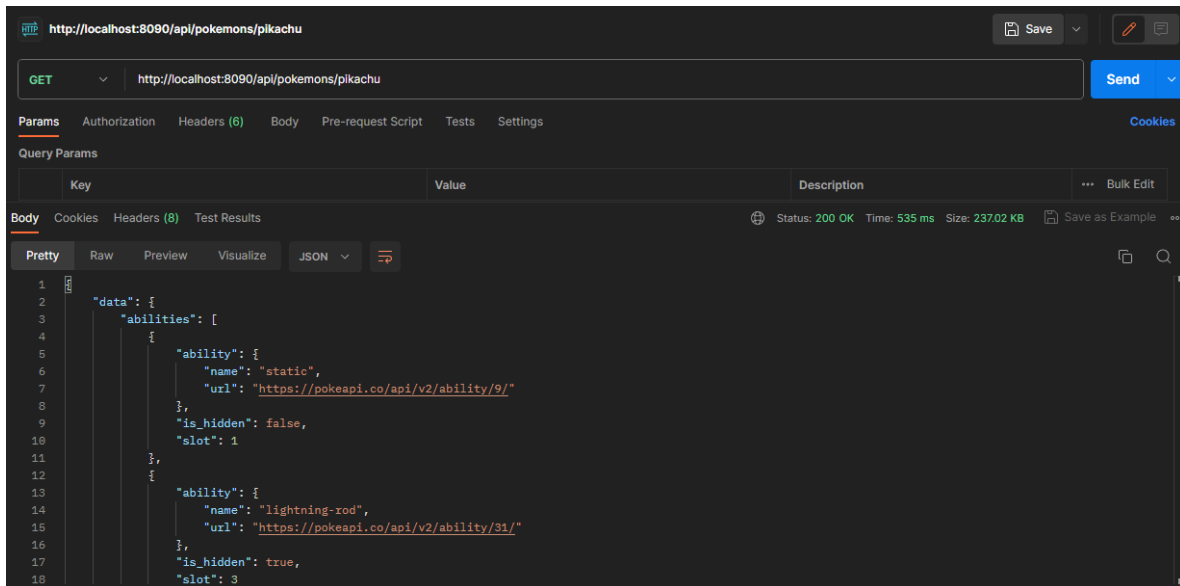


Ilustración 8. Petición GET, para mostrar Pokémon por nombre: • <http://localhost:8090/api/pokemons/pikachu>

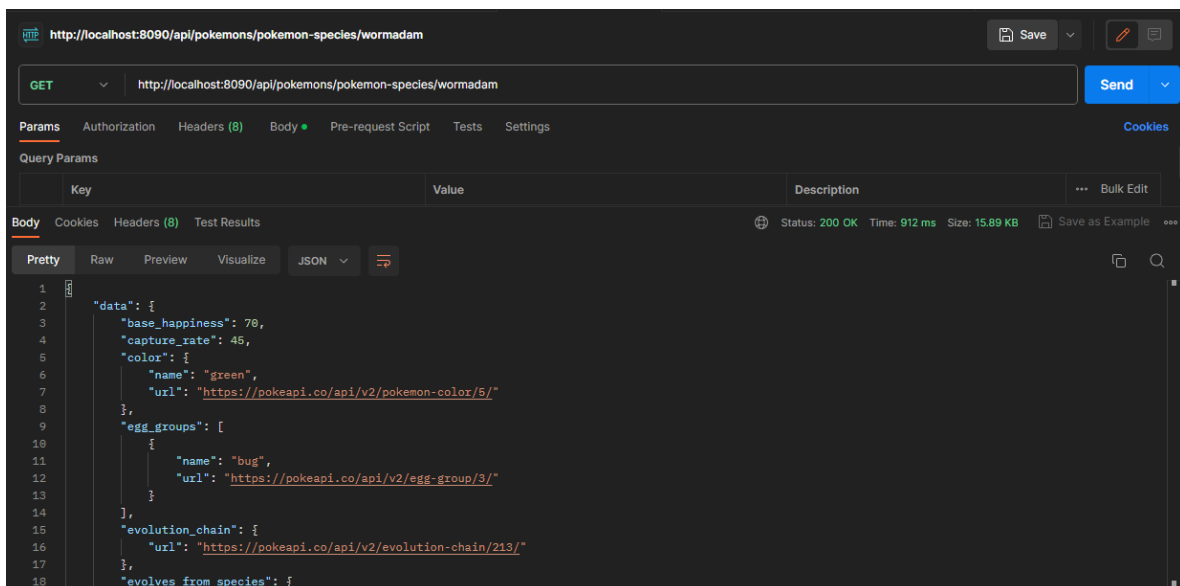


Ilustración 9. Petición GET, para mostrar Pokémon por especie: • <http://localhost:8090/api/pokemons/pokemon-species/wormadam>

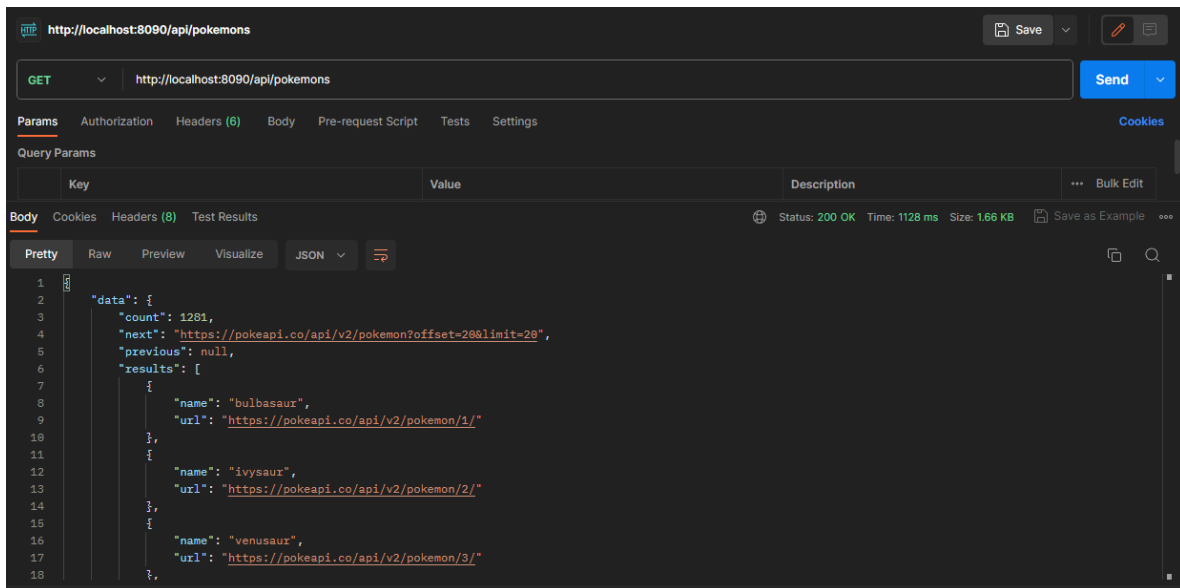


Ilustración 10. Petición GET, para mostrar todos los Pokémon: • <http://localhost:8090/api/pokemons>

6. Despliegue de aplicación en la nube, mediante el servicio de render.

Para este proyecto, se optado por utilizar Render, un servicio en la nube que permite crear y ejecutar aplicaciones y servicios web. Se definen rutas similares a las mencionadas en el punto 3, a continuación se muestra el despliegue de la aplicación.

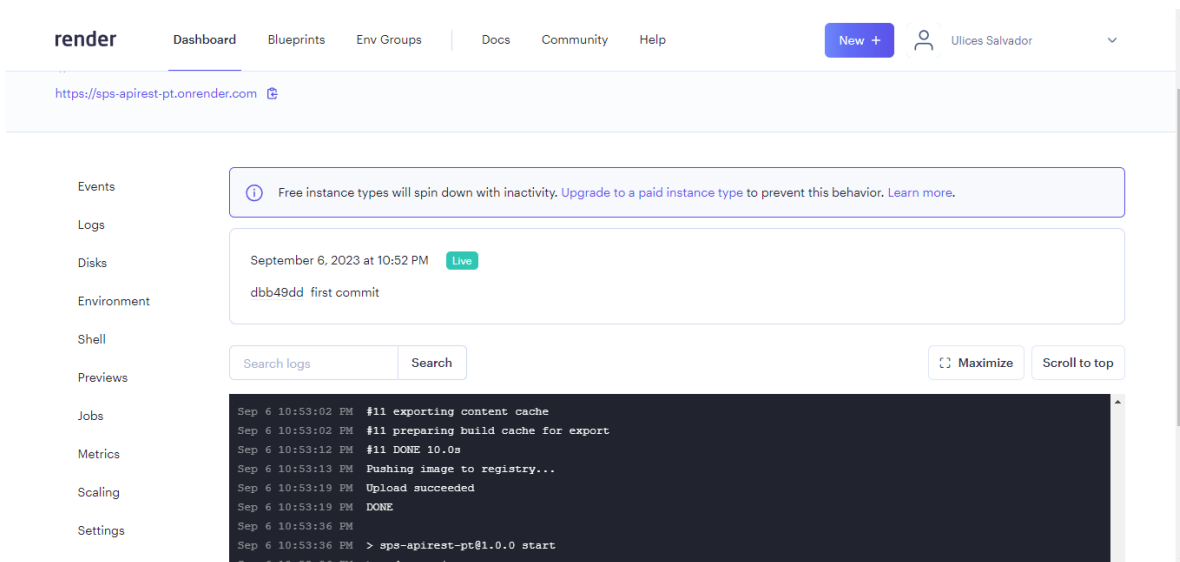


Ilustración 11. Despliegue de aplicación en servicio en la nube de Render.

Para esta aplicación se definen diferentes rutas, similares a las del punto 3.

En la que se muestran las siguientes:

- <https://sps-apirest-pt.onrender.com/>



Acceso denegado

Ilustración 12. Despliegue en render: Directorio público "Index.js"

- <https://sps-apirest-pt.onrender.com/api/pokemons/>

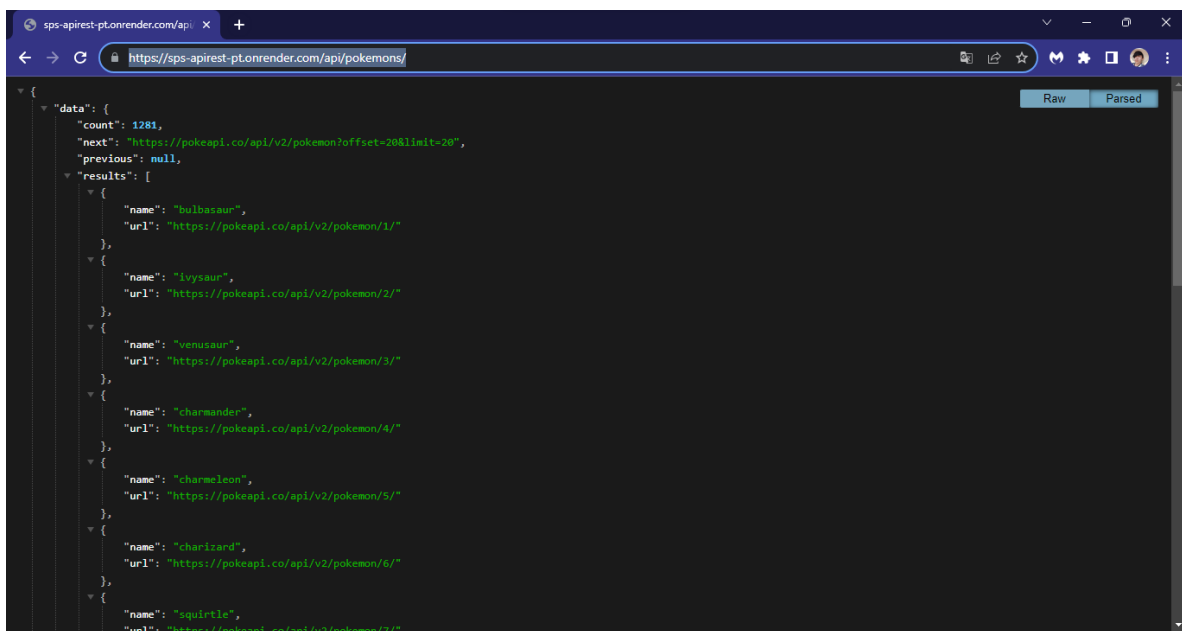


Ilustración 13. Despliegue en render: Mostrar todos los Pokémon.

- <https://sps-apirest-pt.onrender.com/api/pokemons/pikachu>
[https://sps-apirest-pt.onrender.com/api/pokemons/\[namePokemon\]](https://sps-apirest-pt.onrender.com/api/pokemons/[namePokemon])

```

{
  "data": {
    "abilities": [
      {
        "ability": {
          "name": "static",
          "url": "https://pokeapi.co/api/v2/ability/9/"
        },
        "is_hidden": false,
        "slot": 1
      },
      {
        "ability": {
          "name": "lightning-rod",
          "url": "https://pokeapi.co/api/v2/ability/31/"
        },
        "is_hidden": true,
        "slot": 3
      }
    ],
    "base_experience": 112,
    "forms": [
      {
        "name": "pikachu",
        "url": "https://pokeapi.co/api/v2/pokemon-form/25/"
      }
    ],
    "game_indices": [
      {
        "game_index": 84,
        "version": {
          "name": "red",
          "url": "https://pokeapi.co/api/v2/version/1/"
        }
      }
    ]
  }
}

```

Ilustración 14. Despliegue en render: Mostrar Pokémon por nombre.

- <https://sps-apirest-pt.onrender.com/api/pokemons/pokemon-species/wormadam>
[https://sps-apirest-pt.onrender.com/api/pokemons/pokemon-species/\[nameSpecies\]](https://sps-apirest-pt.onrender.com/api/pokemons/pokemon-species/[nameSpecies])

```

{
  "data": {
    "base_happiness": 70,
    "capture_rate": 45,
    "color": {
      "name": "green",
      "url": "https://pokeapi.co/api/v2/pokemon-color/5/"
    },
    "egg_groups": [
      {
        "name": "bug",
        "url": "https://pokeapi.co/api/v2/egg-group/3/"
      }
    ],
    "evolution_chain": {
      "url": "https://pokeapi.co/api/v2/evolution-chain/213/"
    },
    "evolves_from_species": {
      "name": "burmy",
      "url": "https://pokeapi.co/api/v2/pokemon-species/412/"
    },
    "flavor_text_entries": [
      {
        "flavor_text": "When BURMY evolved, its cloak became a part of this Pokémon's body. The cloak is never shed.",
        "language": {
          "name": "en",
          "url": "https://pokeapi.co/api/v2/language/9/"
        },
        "version": {
          "name": "diamond",
          "url": "https://pokeapi.co/api/v2/version/12/"
        }
      }
    ]
  }
}

```

Ilustración 15. Despliegue en render: Mostrar Pokémon por especie.