PokeApi

Tabla de contenidos

Introducción

PokeApi es una API desarrollada en Python utilizando el framework Flask que permite obtener información sobre los Pokémon. La API cuenta con un esquema de autenticación y autorización para garantizar la seguridad de los datos.

Funcionamiento

La API está desarrollada en Python utilizando el framework Flask para crear los endpoints necesarios. También utiliza Flask-JWT-Extended para manejar la autenticación con tokens JWT y Flask-SQLAlchemy para manejar la base de datos de usuarios. La API utiliza la API de https://pokeapi.co/ para obtener información sobre los Pokémon y devolver los resultados en formato JSON.

Uso de la API

Para probar la API, se puede descargar la imagen Dockerizada desde Docker Hub utilizando el comando `docker pull jguilartegg/poke_apiv3`. Luego, se ejecuta la imagen utilizando el comando `docker run -p 5000:5000 jguilartegg/poke_apiv3` para iniciar la aplicación en el puerto 5000.

Una vez que la aplicación esté en ejecución, se puede interactuar con la API utilizando una herramienta como Postman. Para hacerlo, debes enviar solicitudes HTTP a los diferentes endpoints de la API, especificando el método HTTP adecuado (GET, POST, DELETE, etc.) y proporcionando los datos necesarios en el cuerpo de la solicitud o en los parámetros de la URL.

Cómo usar los endpoints con Postman

Para usar los endpoints de la API con Postman, se deben seguir estos pasos:

1. Abre Postman y crea una nueva solicitud.
2. Selecciona el método HTTP adecuado (GET, POST, DELETE, etc.) según el endpoint que deseas utilizar.
3. Ingresa la URL del endpoint en el campo “Enter request URL”, reemplazando <host> con la dirección IP o el nombre de host donde se está ejecutando la aplicación y <port> con el puerto donde se está ejecutando (por defecto es 5000). Por ejemplo, si estás ejecutando la aplicación en tu computadora local, puedes ingresar http://localhost:5000/login para utilizar el endpoint de inicio de sesión.
4. Si el endpoint requiere datos en el cuerpo de la solicitud (por ejemplo, el endpoint de inicio de sesión requiere un username y un password), selecciona la pestaña “Body” y luego selecciona “raw” y “JSON” en las opciones desplegables. Luego, ingresa los datos necesarios en formato JSON en el campo de texto.
5. Si el endpoint requiere un token de acceso JWT (todos los endpoints excepto /login), selecciona la pestaña “Headers” y agrega un nuevo encabezado con el nombre “Authorization” y el valor “Bearer <token>”, reemplazando <token> con el token de acceso obtenido al iniciar sesión.
6. Envia la solicitud a la API. Los resultados se mostrarán en la sección “Response”.

A continuación se muestra una descripción detallada de cada uno de los endpoints disponibles en la API junto con ejemplos concretos de solicitudes y respuestas:

Iniciar sesión

Para iniciar sesión en la API, se debe enviar una solicitud POST al endpoint /login con un cuerpo en formato JSON que contenga el username y password. Por ejemplo:

```
POST http://localhost:5000/login
Content-Type: application/json

{
   "username": "tu-usuario",
   "password": "tu-contraseña"
}
```

Si las credenciales son correctas, la API devolverá un token de acceso JWT en formato JSON:

```
{
   "access_token": "tu-token-de-acceso"
}
```

Debes incluir este token de acceso en el encabezado “Authorization” de las solicitudes posteriores a los endpoints protegidos de la API.

Registrar un nuevo usuario

Para registrar un nuevo usuario en la base de datos, debes enviar una solicitud POST al endpoint `/register` con un cuerpo en formato JSON que contenga el `username` y `password` del nuevo usuario. También debes incluir el token de acceso JWT en el encabezado “Authorization” de la solicitud. Por ejemplo:

```
POST http://localhost:5000/register
Content-Type: application/json
Authorization: Bearer tu-token-de-acceso

{
   "username": "nuevo-usuario",
   "password": "nueva-contraseña"
}
```

Si el registro se realiza correctamente, la API devolverá un mensaje de éxito en formato JSON:

```
{
   "msg": "Usuario registrado con éxito"
}
```

Listar usuarios

Para listar todos los usuarios registrados en la base de datos, debes enviar una solicitud GET al endpoint `/list_users`. También debes incluir el token de acceso JWT en el encabezado “Authorization” de la solicitud. Por ejemplo:

```
GET http://localhost:5000/list_users
Authorization: Bearer tu-token-de-acceso
```

Si tienes permisos para realizar esta acción, la API devolverá una lista de todos los usuarios en formato JSON:

```
[
   {
      "username": "usuario1",
      "is_admin": false
   },
   {
      "username": "usuario2",
      "is_admin": true
   },
   ...
]
```

Eliminar un usuario

Para eliminar a un usuario especificado por su `username`, debes enviar una solicitud DELETE al endpoint `/delete_user` con un cuerpo en formato JSON que contenga el `username` del usuario a eliminar. También debes incluir el token de acceso JWT en el encabezado “Authorization” de la solicitud. Por ejemplo:

```
DELETE http://localhost:5000/delete_user
Content-Type: application/json
Authorization: Bearer tu-token-de-acceso

{
   "username": "usuario-a-eliminar"
}
```

Si tienes permisos para realizar esta acción y el usuario existe, la API devolverá un mensaje de éxito en formato JSON:

```
{
   "msg": "Usuario eliminado con éxito"
}
```

Obtener los tipos de un Pokémon

Para obtener los tipos de un Pokémon especificado por su nombre, debes enviar una solicitud GET al endpoint `/pokemon/type/<name>`, reemplazando `<name>` con el nombre del Pokémon. También debes incluir el token de acceso JWT en el encabezado “Authorization” de la solicitud. Por ejemplo:

```
GET http://localhost:5000/pokemon/type/pikachu
Authorization: Bearer tu-token-de-acceso
```

La API devolverá una lista con los tipos del Pokémon en formato JSON:

```
[
   "electric"
]
```

Obtener un Pokémon aleatorio de un tipo específico

Para obtener un Pokémon aleatorio de un tipo específico, debes enviar una solicitud GET al endpoint /pokemon/random/<type>, reemplazando <type> con el tipo deseado. También debes incluir el token de acceso JWT en el encabezado “Authorization” de la solicitud. Por ejemplo:

```
GET http://localhost:5000/pokemon/random/fire
Authorization: Bearer tu-token-de-acceso
```

La API devolverá el nombre del Pokémon aleatorio seleccionado en formato JSON:

```
"charmander"
```

Obtener el Pokémon con el nombre más largo de un tipo específico

Para obtener el Pokémon con el nombre más largo de un tipo específico, debes enviar una solicitud GET al endpoint /pokemon/longest-name/<type>, reemplazando <type> con el tipo deseado. También debes incluir el token de acceso JWT en el encabezado “Authorization” de la solicitud. Por ejemplo:

```
GET http://localhost:5000/pokemon/longest-name/water
Authorization: Bearer tu-token-de-acceso
```

La API devolverá el nombre del Pokémon con el nombre más largo en formato JSON:

```
"blastoise"
```

Esquema de autenticación y autorización

La API cuenta con un esquema de autenticación y autorización basado en tokens JWT (JSON Web Tokens). Para acceder a los endpoints protegidos de la API (todos excepto /login y /register), el usuario debe iniciar sesión utilizando el endpoint /login y proporcionar un username y un password válidos. Si las credenciales son correctas, el endpoint devuelve un token de acceso JWT que debe incluirse en el encabezado “Authorization” de las solicitudes posteriores a los endpoints protegidos.

Además, algunos endpoints (como /register, /list_users y /delete_user) requieren que el usuario que realiza la solicitud sea un administrador. Esto se verifica utilizando el campo is_admin del usuario en la base de datos.