2023

DOCUMENTACIÓN API: ORGANIZADOR DE PERSONAJES

PROYECTO SEGUNDO TRIMESTRE PROGRAMACIÓN DE SERVICIOS Y PROCESOS

DIEGO EMANUEL SALINAS MARTINEZ

Contenido

Introducción

La API que se presenta tiene como finalidad principal la gestión de los personajes de una historia que desees crear, independientemente del medio en el que se vaya a plasmar, ya sea un videojuego, una novela, una película u otro.

En esta versión de la API, encontrarás una amplia gama de funcionalidades que te permitirán registrar de manera sencilla y eficiente todos los aspectos relevantes de tus personajes, como su género, especie y localización. De esta forma, podrás tener un mayor control y orden en el desarrollo de tu obra, facilitando su creación y garantizando una experiencia de usuario más satisfactoria.

En esta versión de la API podrás registrar personajes, géneros, especies y localizaciones. Todos los anteriormente mencionados tienen sus propios métodos GET, PUT, POST y DELETE. Estos datos están entrelazados y hay algunas dependencias que se explicarán más adelante.

Requisitos

Configurar la base de datos

Para configurar correctamente la base de datos, dirígete a la carpeta "app" ubicada en el directorio principal del proyecto y abre el archivo config.py. En ella, encontrarás toda la información necesaria para su configuración y el modo de ejecución de la aplicación.

En particular, deberás ingresar los datos de tu base de datos y definir si deseas que se ejecute en modo depuración o no. Te recomendamos observar el ejemplo proporcionado para tener una idea más clara de cómo realizar la configuración.

La estructura a seguir es la siguiente:

```
LA estructura es la siguiente
class DevelopmentConfig():

    DEBUG=True
    #Configuracion base de datos
    MYSQL_HOST='localhost'
    MYSQL_USER='root'
    MYSQL_PASSWORD = ''
    MYSQL_DB='api_personajes'

config={
    'development':DevelopmentConfig
}
```

En este bloque de código, MYSQL_HOST corresponde a la dirección de tu base de datos, MYSQL_USER es el nombre de usuario asignado para la base de datos, MYSQL_PASSWORD es la contraseña del usuario anteriormente mencionado y MYSQL_DB es el nombre de la base de datos que se utilizará para la API.

Base de datos MySQL

En cuanto a la base de datos, resulta imprescindible contar con un sistema de gestión de datos como MySQL o MariaDB para almacenar los personajes de tu historia de forma organizada y accesible.

Para ello, en el directorio raíz del proyecto, podrás encontrar la carpeta "Base de datos" que contiene la estructura necesaria para su implementación. En caso de no localizarla, te recomendamos descargar el proyecto desde el enlace principal o contactar al creador para obtenerla.

Una vez accedas a la carpeta, deberás importar el archivo .sql en la base de datos asignada a la API para poder comenzar a utilizarla.

Librerías

Por otro lado, todas las librerías requeridas para el correcto funcionamiento de la API se encuentran en el archivo "requirements.txt" que se ubica en la carpeta principal del proyecto. Asegúrate de instalar todas las librerías enlistadas para garantizar su correcta operación.

Configuración de CORS

Para habilitar el acceso CORS en la aplicación para tu dominio, es necesario realizar una modificación en el archivo application.py que se encuentra en el directorio app de la raíz del

proyecto. Puedes ver la estructura de código que se encuentra al final del documento para saber exactamente en qué parte de archivo se encuentra la línea de código.

Actualmente, la línea de código que habilita CORS está limitando el acceso a la aplicación solo desde el origen "localhost". El parámetro "app" se refiere a la instancia de la aplicación Flask en la que se habilitará CORS, mientras que el parámetro "resources" especifica los recursos que se permitirán, utilizando la expresión regular "/*" para permitir el acceso a cualquier recurso dentro de la aplicación.

```
CORS(app, resources={r"/*":{"origins":'localhost'}})
```

Para habilitar el acceso a otros dominios, se debe modificar la línea de código especificando el dominio correspondiente en el parámetro "origins". Por ejemplo, para permitir el acceso al dominio "www.mycomputer.es" a todos los recursos de la aplicación, se debe modificar la línea de código de la siguiente manera:

```
CORS(app, resources={r"/*":{"origins":'www.mycomputer.es'}})
```

De esta forma, se habilita el acceso desde el origen especificado a todos los recursos de la aplicación utilizando la expresión regular "/*".

Personajes

Esquema

Clave	Tipo	Descripción
id	int	ID del personaje
Nombre	string	Nombre del personaje
Apellido	string	Apellidos del personaje
Edad	int	La edad del personaje en el momento en el que lo indicas
Descripcion	string	Pequeña descripción del personaje para tener una pequeña idea sobre él
Padre	int	ID del personaje del padre o -1 si el padre no está registrado
Madre	int	ID del personaje de la madre o -1 si la madre no está registrada
especie	int	ID de la especie
genero	int	ID del género
imagen	String (url)	Imagen del personaje, es una imagen de la cara del personaje
nacimiento	int	ID del lugar de nacimiento del personaje
aparicion	int	ID del lugar en donde se ha visto por primera vez al personaje
localizacion	int	ID del lugar en donde se encuentra el personaje actualmente o donde se le ha visto por última vez

Método GET

URL

/personajes

Parámetros de consulta

1. **id (opcional):** uno o varios identificadores de personajes separados por coma (solo números).
2. **nombre (opcional):** uno o varios nombres de personajes separados por coma.
3. **apellido (opcional):** uno o varios apellidos de personajes separados por coma.
4. **edad (opcional):** uno o varios valores de edad separados por coma (solo números).
5. **padre (opcional):** uno o varios identificadores de personajes que sean padres separados por coma (solo números).
6. **madre (opcional):** uno o varios identificadores de personajes que sean madres separados por coma (solo números).
7. **especie (opcional):** uno o varios identificadores de especies separados por coma (solo números).
8. **genero (opcional):** uno o varios identificadores de géneros separados por coma (solo números).
9. **nacimiento (opcional):** uno o varios identificadores de lugares de nacimiento separados por coma (solo números).
10. **localizacion (opcional):** uno o varios identificadores de lugares de localización separados por coma (solo números).
11. **aparicion (opcional):** uno o varios identificadores de episodios de aparición separados por coma (solo números).

Respuestas

1. Si la consulta es exitosa, la API devolverá un objeto JSON que contiene una lista de personajes y un mensaje indicando que los personajes han sido listados. La lista de personajes puede estar vacía si has indicado un filtro y ningún personaje lo cumple o no se ha registrado ningún personaje.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.

Ejemplo de uso

/personajes?id=1,2&nombre=Rick&apellido=Sánchez&edad=70&genero=1&aparicion=1,2

Método POST

URL

/personajes

Parámetros de consulta

1. **nombre (Obligatorio):** Nombre del personaje.
2. **apellidos (Obligatorio):** Apellido del personaje.
3. **edad (Obligatorio):** Edad del personaje (solo números).
4. **descripcion (Obligatorio):** Descripción del personake
5. **padre (Obligatorio):** ID de un personaje existente en la base de datos, en caso de no existir la base de datos correspondiente y no es igual a -1 lanza error (solo números).
6. **madre (Obligatorio):** ID de un personaje existente en la base de datos, en caso de no existir la base de datos correspondiente y no es igual a -1 lanza error (solo números).
7. **especie (Obligatorio):** ID de una especie existente en la base de datos, en caso de no existir la especie en su tabla correspondiente lanza error (solo números).
8. **genero (Obligatorio):** ID de un género existente en la base de datos, en caso de no existir el género en su tabla correspondiente lanza error (solo números).
9. **imagen (Obligatorio):** Url de la imagen del personaje.
10. **nacimiento (Obligatorio):** ID de una localización existente en la base de datos, en caso de no existir la localización lanza error (solo números).
11. **localizacion (Obligatorio):** ID de una localización existente en la base de datos, en caso de no existir la localización lanza error (solo números).
12. **aparicion (obligatorio):** ID de una localización existente en la base de datos, en caso de no existir la localización lanza error (solo números).

La API está preparada para recibir estos datos como JSON y desde un formulario.

Respuestas

1. Si la consulta es exitosa, la API devolverá un mensaje indicando que el personaje ha sido registrado.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.
    a. Error que indica que ha faltado un dato, el dato pasado es nulo, el dato pasado no es nulo, pero es cadena vacía, el dato pasado es valor numérico y es negativo y no es del campo Padre o Madre, y/o es valor numérico y es igual a 0.
    Devuelve un mensaje con el listado de parámetro que da error. Un ejemplo: Le paso un formulario que no contiene apellido y Madre, y la aparición con el campo vacío.
    {
    "mensaje": "No se ha podido registrar el personaje ya que los siguientes valores son nulos, cadena vacía y/o no cumplen con los requisitos. Compruebe estos parametros y vuelva a intentarlo: Apellidos,Madre,Aparicion."
    }
    b. Error al insertar un valor que dependa de otra tabla en la base de datos, un ejemplo sería la localización ya que está debe de existir para que puedas ser asignado. Estos valores son padre (a no ser que sea -1), madre (a no ser que sea -1), especie, género y aparición, nacimiento y localización. Ejemplo:
    Inserto una localización que no existe a la localización de aparición:
    {

"mensaje": "No se puede asignar el/la aparicion debido a que este dato depende de su existencia en la base de datos, ya sea de la misma tabla u otra de la que dependa."
}

c. Otros errores.

Método PUT

URL

/personajes/<ID>

Parámetros de consulta

En la URL tiene que introducir el ID del personaje que quiere eliminar. Por ejemplo, si quiere eliminar al personaje con ID 3 el enlace sería el siguiente.

/personajes/3

La API está preparada para recibir estos datos de modificación como JSON y desde un formulario.

1. **nombre (opcional):** Nombre del personaje.
2. **apellidos (opcional):** Apellido del personaje.
3. **edad (Opcional):** Edad del personaje (solo números).
4. **descripcion (Opcional):** Descripción del personake
5. **padre (Opcional):** ID de un personaje existente en la base de datos, en caso de no existir la base de datos correspondiente y no es igual a -1 lanza error (solo números).
6. **madre (Opcional):** ID de un personaje existente en la base de datos, en caso de no existir la base de datos correspondiente y no es igual a -1 lanza error (solo números).
7. **especie (Opcional):** ID de una especie existente en la base de datos, en caso de no existir la especie en su tabla correspondiente lanza error (solo números).
8. **genero (Opcional):** ID de un género existente en la base de datos, en caso de no existir el género en su tabla correspondiente lanza error (solo números).
9. **imagen (Opcional):** Url de la imagen del personaje.
10. **nacimiento (Opcional):** ID de una localización existente en la base de datos, en caso de no existir la localización lanza error (solo números).
11. **localizacion (Opcional):** ID de una localización existente en la base de datos, en caso de no existir la localización lanza error (solo números).
12. **aparicion (opcional):** ID de una localización existente en la base de datos, en caso de no existir la localización lanza error (solo números).

Respuestas

1. Si la consulta es exitosa, la API devolverá un mensaje indicando que el personaje ha sido modificado.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.
    a. Error que indica el dato pasado es nulo, el dato pasado no es nulo, pero es cadena vacía, el dato pasado es valor numérico y es negativo y no es del campo Padre o Madre, y/o es valor numérico y es igual a 0.
    Devuelve un mensaje con el listado de parámetro que da error. Un ejemplo: Le paso un formulario con un parámetro como la aparición con el campo vacío.

    {
    "mensaje": "No se ha podido registrar el personaje ya que los siguientes valores son nulos, cadena vacía y/o no cumplen con los requisitos. Compruebe estos parametros y vuelva a intentarlo: Aparicion."
    }

    b. Error al insertar un valor que dependa de otra tabla en la base de datos, un ejemplo sería la localización ya que está debe de existir para que puedas ser

asignado. Estos valores son padre (a no ser que sea -1), madre (a no ser que sea -1), especie, género y aparición, nacimiento y localización. Ejemplo:
Inserto una localización que no existe a la localización de aparición:
{
"mensaje": "No se puede asignar el/la aparicion debido a que este dato depende de su existencia en la base de datos, ya sea de la misma tabla u otra de la que dependa."
}

c. Si no manda ningún dato para modificar ya sea por json o formulario devuelve el siguiente mensaje:
{
"mensaje": "Tiene que insertar mínimo un datos para poder modificar al personaje, inténtelo de nuevo introduciendo un dato modificable.\nEntre los datos modificables tenemos nombre,apellidos,edad,descripcion,padre,madre,especie,imagen,nacimiento,localizacion,aparicion."
}

d. Error del que el ID introducido de un personaje no este relacionado con ningún personaje.
{
"mensaje": "No existe ningún personaje con el id que ha especificado. Por favor, verifique que el id esté relacionado un personaje existente y vuelva a intentarlo."
}

e. Has introducido un como id un valor no numérico:
{
"mensaje": "El id que tienes que introducir tiene que ser un valor numérico."
}

f. Otros errores.

Método DELETE

URL

/personajes/<ID>

Parámetros de consulta

- **id (Obligatorio):** Identificador del personaje que va a eliminar, va en el enlace (solo números).

Respuestas

1. Si la consulta es exitosa, la API devolverá un mensaje indicando que el personaje ha sido eliminado.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.
    1. Error del que el ID introducido de un personaje no este relacionado con ningún personaje.
    {
    "mensaje": "No existe ningún personaje con el id que ha especificado. Por favor, verifique que el id esté relacionado un personaje existente y vuelva a intentarlo."
    }
    2. Error de que el personaje que trata de eliminar está como el padre o madre de un personaje.
    {

"mensaje": "No se puede eliminar el personaje ya que es padre o madre de un personaje, modifique los personajes ante de volver a intentarlo."
}

3. Has introducido un como id un valor no numérico:
   {
   "mensaje": "El id que tienes que introducir tiene que ser un valor numérico."
   }
4. Otros errores.

Ejemplo de uso

Eliminación del personaje con ID 6: /personajes/6

Localizaciones

Esquema

Clave	Tipo	Descripción
id	int	ID de la localización
Coordenadas	string	Coordenadas de la localización en caso de querer introducirla
Ciudad	string	Nombre de la ciudad en caso de que exista en dicha dimensión
País	string	Nombre del país, puede ser también el de un reino en caso de ser un mundo de fantasía
Descripción	string	Descripción de la localización, puedes rellenar con datos que han pasado o que se tiene que tener en cuenta, etc.
Dimensión	string	Dimensión en la que pertenece en caso de querer añadir más de una dimensión. Como mínimo debe de haber una dimensión.
Población	int	Cantidad de habitantes que tiene el sitio
Moneda	string	Nombre de la moneda que utiliza en esa localización

Método GET

URL

/localizaciones

Parámetros de consulta

1. **id (opcional):** uno o varios identificadores de localizaciones separados por coma (solo números).
2. **ciudad (opcional):** una o varias ciudades separados por coma.
3. **pais (opcional):** uno o varios países separados por coma.
4. **dimension (opcional):** una o varias dimensiones separados por coma.
5. **poblacion (opcional):** una cantidad de habitantes de una localización (solo números).
6. **moneda (opcional):** una o varias monedas separados por coma.

Respuestas

1. Si la consulta es exitosa, la API devolverá un objeto JSON que contiene una lista de localizaciones y un mensaje indicando que los personajes han sido listados. La lista de localizaciones puede estar vacía si has indicado un filtro y ninguna localización lo cumple o no se ha registrado ninguna localización.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.

Ejemplo de uso

/localizaciones?dimension=c-342,flauta43&poblacion=3424234&moneda=euro,dólar

Método POST

URL

/localizaciones

Parámetros de consulta

1. **coordenadas (Obligatorio):** Coordenadas de la localización.
2. **ciudad (Obligatorio):** Nombre de la ciudad.
3. **pais (Obligatirio):** Nombre del país.
4. **descripcion (Obligatorio):** Descripción de la localización.
5. **dimension (Obligatorio):** Nombre de una dimensión.
6. **poblacion (Obligatorio):** Cantidad de habitantes de esa localización.
7. **moneda (Obligatorio):** Nombre de la moneda que se utiliza.

La API está preparada para recibir estos datos como JSON y desde un formulario.

Respuestas

1. Si la consulta es exitosa, la API devolverá un mensaje indicando que la localización ha sido registrada.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.
    1. Error que indica que ha faltado un dato, el dato pasado es nulo, el dato pasado no es nulo, pero es cadena vacía, el dato pasado es valor numérico y es negativo, y/o es valor numérico y es igual a 0.
       Devuelve un mensaje con el listado de parámetro que da error. Un ejemplo: Le paso un formulario que no contiene la moneda.
       {
       "mensaje": "No se ha podido registrar la localización ya que los siguientes valores son nulos, cadena vacía y/o no cumplen con los requisitos. Compruebe estos parámetros y vuelva a intentarlo: Moneda."
       }
    2. Otros errores.

Método PUT

URL

/localizaciones/<ID>

Parámetros de consulta

En la URL tiene que introducir el ID de la localización que quiere modificar. Por ejemplo, si quiere modificar la localización con ID 3 el enlace sería el siguiente.

/localizaciones/3

La API está preparada para recibir estos datos de modificación como JSON y desde un formulario, a excepción del ID:

1. **id (Obligatorio):** Identificador del personaje que va a eliminar, va en el enlace (solo números).
2. **coordenada(opcional):** Coordenada de la localización.
3. **ciudad (opcional):** nombre de la ciudad.
4. **pais (opcional):** Nombre del país.
5. **dimension (opcional):** Nombre de la dimensión.
6. **descripcion (opcional):** Descripción de la localización.
7. **poblacion (opcional):** una cantidad de habitantes de una localización (solo números).
8. **moneda (opcional):** Nombre de la moneda.

Respuestas

1. Si la consulta es exitosa, la API devolverá un mensaje indicando que la localización ha sido modificada.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.
    a. Error que indica el dato pasado es nulo, el dato pasado no es nulo, pero es cadena vacía, el dato pasado es valor numérico y es negativo y/o es valor numérico y es igual a 0.
       Devuelve un mensaje con el listado de parámetro que da error. Un ejemplo: Le paso un formulario con un parámetro como la población con el campo vacío.
       {
       "mensaje": "Los siguientes valores que has pasado son cadena vacía o en caso de ser valor numerico son negativos: Poblacion."
       }

b. Si no manda ningún dato para modificar ya sea por json o formulario devuelve el siguiente mensaje:
{
"mensaje": "Tiene que insertar mínimo un dato para poder modificar la localización, inténtelo de nuevo introduciendo un dato modificable.\nEntre los datos modificables tenemos la coordenada, la ciudad, el pais, la descripcion, la dimension, la población y la moneda."
}
c. El ID de la localización que desea modificar no existe:
{
"mensaje": "No existe ninguna localización con el id que ha especificado. Por favor, verifique que el id esté relacionado a una especie existente y vuelva a intentarlo."
}
d. Has introducido un como id un valor no numérico:
{
"mensaje": "El id que tienes que introducir tiene que ser un valor numérico."
}
e. Otros errores.

Método DELETE

URL

/localizaciones/<ID>

Parámetros de consulta

- **id (Obligatorio):** Identificador del personaje que va a eliminar, va en el enlace (solo números).

Respuestas

1. Si la consulta es exitosa, la API devolverá un mensaje indicando que la localización ha sido eliminada.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.
    1. Error del que el ID introducido de una localización no esté relacionado con ninguna localización.
    {
    "mensaje": "No existe ninguna localización con el id que ha especificado. Por favor, verifique que el id esté relacionado una localización existente y vuelva a intentarlo."
    }
    2. Has introducido un como id un valor no numérico:
    {
    "mensaje": "El id que tienes que introducir tiene que ser un valor numérico."
    }
    3. La localización que intentas eliminar la tiene asignada un personaje:

    {

    "mensaje": "La localización que quiere eliminar lo tiene asignado un personaje. Por favor, asigne otro género a los personajes que lo tienen asignado y luego vuelva a intentarlo."

    }

4. Otros errores.

Ejemplo de uso

Eliminación de la localización con ID 6: /localizaciones/6

Géneros

Esquema

Clave	Tipo	Descripción
ID	int	ID del género
nombre	string	Nombre del género
Descripción	string	Descripción del género

Método GET

URL

/generos

Parámetros de consulta

1. **id (opcional):** uno o varios identificadores de géneros separados por coma (solo números).
2. **nombre (opcional):** uno o varios nombres de géneros separados por coma.

Respuestas

1. Si la consulta es exitosa, la API devolverá un objeto JSON que contiene una lista de géneros y un mensaje indicando que los géneros han sido listados. La lista de géneros puede estar vacía si has indicado un filtro y ningún género lo cumple o no se ha registrado ningún género.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.

Ejemplo de uso

/generos?nombre=hombre,mujer&id=3,4,5,2,45,34

Método POST

URL

/generos

Parámetros de consulta

1. **descripcion (Obligatorio):** la descripción de género.
2. **nombre (Obligatorio):** Nombres del género.

La API está preparada para recibir estos datos como JSON y desde un formulario.

Respuestas

1. Si la consulta es exitosa, la API devolverá un mensaje indicando que el género ha sido registrado.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.
    1. Error que indica que ha faltado un dato, el dato pasado es nulo, el dato pasado no es nulo, pero es cadena vacía, el dato pasado es valor numérico y es negativo, y/o es valor numérico y es igual a 0.
       Devuelve un mensaje con el listado de parámetro que da error. Un ejemplo: Le paso un formulario que no contiene la Descripción.
       {
       "mensaje": "No se ha podido registrar el género ya que los siguientes valores son nulos, cadena vacía y/o no cumplen con los requisitos. Compruebe estos parametros y vuelva a intentarlo: Descripcion."
       }
    2. Otros errores.

Método PUT

URL

/generos/<ID>

Parámetros de consulta

En la URL tiene que introducir el ID del género que quiere modificar. Por ejemplo, si quiere modificar un género con ID 3 el enlace sería el siguiente.

/generos/3

La API está preparada para recibir estos datos de modificación como JSON y desde un formulario, a excepción del ID:

1. **descripcion (opcional):** la descripción de género.
2. **nombre (opcional):** Nombres del género.

Pone que las dos son opcionales, pero en realidad una si es obligatoria.

Respuestas

1. Si la consulta es exitosa, la API devolverá un mensaje indicando que el género ha sido modificado.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.
    a. Error que indica el dato pasado es cadena vacía.
    Devuelve un mensaje con el parámetro que da error. Un ejemplo: Le paso un formulario con un parámetro como la descripción con el campo vacío.
    {
    "mensaje": "No se puede modificar el/la descripcion el valor reemplazante es una cadena vacía."
    }
    b. Si no manda ningún dato para modificar ya sea por json o formulario devuelve el siguiente mensaje:
    {
    "mensaje": "Tiene que insertar mínimo un dato para poder modificar el género, inténtelo de nuevo introduciendo un dato modificable.\nEntre los datos modificables tenemos el nombre y la descripcion"
    }
    c. El ID del género que desea modificar no existe:
    {
    "mensaje": "No existe ningún género con el id que ha especificado. Por favor, verifique que el id esté relacionado a un género existente y vuelva a intentarlo."
    }
    d. Has introducido un como id un valor no numérico:
    {
    "mensaje": "El id que tienes que introducir tiene que ser un valor numérico."
    }
    e. Otros errores.

Método DELETE

URL

/generos/<ID>

Parámetros de consulta

- **id (Obligatorio):** Identificador del género que va a eliminar, va en el enlace (solo números).

Respuestas

1. Si la consulta es exitosa, la API devolverá un mensaje indicando que el género ha sido eliminado.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.
    1. Error del que el ID introducido de un género no esté relacionado con ningún género.
       {
       "mensaje": "No existe ningún género con el id que ha especificado. Por favor, verifique que el id esté relacionado a una especie existente y vuelva a intentarlo."
       }
    2. Has introducido un como id un valor no numérico:
       {
       "mensaje": "El id que tienes que introducir tiene que ser un valor numérico."
       }
    3. El género que intentas eliminar la tiene asignada un personaje:

       {

       "mensaje": "El género que quiere eliminar lo tiene asignado un personaje. Por favor, asigne otro género a los personajes que lo tienen asignado y luego vuelva a intentarlo."

       }

    4. Otros errores.

Ejemplo de uso

Eliminación del género con ID 6: /generos/6

Especies

Esquema

Clave	Tipo	Descripción
ID	int	ID de la especie
nombre	string	Nombre de la especie
Descripción	string	Descripción de la especie

Método GET

URL

/especies

Parámetros de consulta

1. **id (opcional):** uno o varios identificadores de especies separados por coma (solo números).
2. **nombre (opcional):** uno o varios nombres de especies separados por coma.

Respuestas

1. Si la consulta es exitosa, la API devolverá un objeto JSON que contiene una lista de especies y un mensaje indicando que las especies han sido listados. La lista de especies puede estar vacía si has indicado un filtro y ninguna especie lo cumple o no se ha registrado ninguna especie.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.

Ejemplo de uso

/especie?nombre=alien&id=3,4,5,2,45,34

Método POST

URL

/especies

Parámetros de consulta

1. **descripcion (Obligatorio):** la descripción de la especie.
2. **nombre (Obligatorio):** Nombres de la especie.

La API está preparada para recibir estos datos como JSON y desde un formulario.

Respuestas

1. Si la consulta es exitosa, la API devolverá un mensaje indicando que la especie ha sido registrado.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.
    1. Error que indica que ha faltado un dato, el dato pasado es nulo, el dato pasado no es nulo, pero es cadena vacía, el dato pasado es valor numérico y es negativo, y/o es valor numérico y es igual a 0.
       Devuelve un mensaje con el listado de parámetro que da error. Un ejemplo: Le paso un formulario que no contiene la Descripción.
       {
       "mensaje": "No se ha podido registrar la especie ya que los siguientes valores son nulos, cadena vacía y/o no cumplen con los requisitos. Compruebe estos parámetros y vuelva a intentarlo: Descripcion."
       }
    2. Otros errores.

Método PUT

URL

/especies/<ID>

Parámetros de consulta

En la URL tiene que introducir el ID de la especie que quiere modificar. Por ejemplo, si quiere modificar un género con ID 3 el enlace sería el siguiente.

/especies/3

La API está preparada para recibir estos datos de modificación como JSON y desde un formulario, a excepción del ID:

1. **descripcion (opcional):** la descripción de la especie.
2. **nombre (opcional):** Nombres de la especie.

Pone que las dos son opcionales, pero en realidad una si es obligatoria.

Respuestas

1. Si la consulta es exitosa, la API devolverá un mensaje indicando que la especie ha sido modificada.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.
    1. Error que indica el dato pasado es cadena vacía.
       Devuelve un mensaje con el parámetro que da error. Un ejemplo: Le paso un formulario con un parámetro como la descripción con el campo vacío.
       {
       "mensaje": "No se puede modificar el/la descripcion el valor reemplazante es una cadena vacía."
       }
    2. Si no manda ningún dato para modificar ya sea por json o formulario devuelve el siguiente mensaje:
       {
       "mensaje": "Tiene que insertar mínimo un dato para poder modificar la especie, inténtelo de nuevo introduciendo un dato modificable.\nEntre los datos modificables tenemos el nombre y la descripcion"
       }
    3. El ID dela especie que desea modificar no existe:
       {
       "mensaje": "No existe ninguna especie con el id que ha especificado. Por favor, verifique que el id esté relacionado a una especie existente y vuelva a intentarlo."
       }
    4. Has introducido un como id un valor no numérico:
       {
       "mensaje": "El id que tienes que introducir tiene que ser un valor numérico."
       }
    5. Otros errores.

Método DELETE

URL

/especies/<ID>

Parámetros de consulta

- **id (Obligatorio):** Identificador del género que va a eliminar, va en el enlace (solo números).

Respuestas

1. Si la consulta es exitosa, la API devolverá un mensaje indicando que la especie ha sido eliminada.
2. Si ocurre algún error durante la consulta, la API devolverá un objeto JSON con un mensaje indicando el error.
    1. Error del que el ID introducido de una especie no esté relacionado con ningún género.
       {
       "mensaje": "No existe ninguna especie con el id que ha especificado. Por favor, verifique que el id esté relacionado a una especie existente y vuelva a intentarlo."
       }
    2. Has introducido un como id un valor no numérico:
       {
       "mensaje": "El id que tienes que introducir tiene que ser un valor numérico."
       }
    3. La especie que intentas eliminar la tiene asignada un personaje:

       {

       "mensaje": "La especie que quiere eliminar lo tiene asignado un personaje. Por favor, asigne otra especie a los personajes que lo tienen asignado y luego vuelva a intentarlo."

       }

    4. Otros errores.

Ejemplo de uso

Eliminación de la especie con ID 6: /especies/6

Estructura del código aplicación

1. Importaciones.
2. Creación del objeto de aplicación de Flask.
3. Configuración de CORS.
4. Método GET:
    - GET Personajes.
    - GET Localizaciones.
    - GET géneros.
    - GET especies.
5. Funciones de apoyo a para los métodos GET:
    - Comprobación de datos pasados y generación de consulta SQL para obtener los personajes que coincidan con los filtros en caso de ponerlos.
    - Comprobación de datos pasados y generación de consulta SQL para obtener las localizaciones que coincidan con los filtros en caso de ponerlos.
    - Comprobación de datos pasados y generación de consulta SQL para obtener los géneros o especies que coincidan con los filtros en caso de ponerlos.
6. Método POST:
    - POST Personajes.
    - POST Localizaciones.
    - POST géneros.
    - POST especies.
7. Funciones de apoyo a para los métodos POST:
    - Comprobación de datos pasados y generación de consulta SQL para crear los personajes en la base de datos.
    - Comprobación de datos pasados y generación de consulta SQL para crear las localizaciones en la base de datos.
    - Comprobación de datos pasados y generación de consulta SQL para crear los géneros o especies en la base de datos.
8. Método PUT:
    - PUT Personajes.
    - PUT Localizaciones.
    - PUT géneros.
    - PUT especies.
9. Funciones de apoyo a para los métodos PUT:
    - Comprobación de datos pasados y generación de consulta SQL para modificar los datos del personaje que está ligado al ID que se ha incluido en el enlace.
    - Comprobación de datos pasados y generación de consulta SQL para modificar los datos de la localización que está ligado al ID que se ha incluido en el enlace.
    - Comprobación de datos pasados y generación de consulta SQL para modificar los datos del género o la especie que está ligado al ID que se ha incluido en enlace.
10. Método DELETE:
    - DELETE Personajes.
    - DELETE Localizaciones.
    - DELETE géneros.
    - DELETE especies.
11. Funciones de apoyo a para los métodos DELETE
    - Comprobación de datos pasados y generación de consulta SQL para eliminar la localización que está ligado al ID que se ha incluido en enlace.
    - Comprobación de datos pasados y generación de consulta SQL para eliminar el género o la especie que está ligado al ID que se ha incluido en enlace.
    - Funciones de apoyo generales (funciones que se utilizan más de uno de los métodos anteriormente mencionados).
12. Comprobación de datos.
13. Consultas a base de datos referentes a id de una tabla.

14. Página de error 404
15. Comprobación de archivo principal, carga de configuración, redirección página 404 e inicio de la aplicación.