Universidad ORT Uruguay

Facultad de Ingeniería

Diseño de Aplicaciones 2
Obligatorio 1:
Documentacion Api

Juliette Ruchel - 203942
Francisco Martinez - 233126

Docente: `Ignacio Valle`

Entregado como requisito de la materia Diseño de Aplicaciones 2

https://github.com/ORT-DA2/203942-233126

14 de mayo de 2020

Declaraciones de autoría

Nosotros, Juliette Ruchel y Francisco Martinez, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos Diseño de aplicaciones 2;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Resumen

El presente documento tiene como objetivo especificar la Web Api desarrollada y justificar las decisiones tomadas en lo que respecta a su desarrollo.

Índice general

1. Estándares

La aplicación fue creada siguiendo los estándares establecidos por REST, un estilo de arquitectura de aplicaciones propuesto en el año 2000 por Roy Fielding.[2] Esto consiste en:

- Usa una interfaz uniforme, que ayuda a desacoplar las implementaciones del cliente y el servidor.
- Permite organizar la Api según los recursos del sistema. Por ejemplo, en esta Api se crearon cuatro controllers basándonos en recursos existentes del sistema: Session, CitizenRequest, Type, y Administrator.
- Las urls se deben basar en sustantivos y no verbos.
  Por ejemplo */api/CitizenRequest/2*
- Una colección es un recurso separado del item dentro de la colección, y debería tener su propia URI.
- Se utilizaron los métodos comunes a la mayoría de las Apis REST : GET, PUT, POST & DELETE
- Se utilizaron códigos de status HTTP para indicar el estado de la respuesta al usuario

2. Mecanismos de Autenticación

Además de un mecanismo de autenticación básico como es el login, se agregó un filtro para controlar que el administrador que intenta acceder a los endpoints del sistema tenga acceso.

2.1. Authorization Filter

Verifica que quienes quieren envían una solicitud a un endpoint cumplan ciertas condiciones. Las condiciones son :

- Enviar un token en el header indicando que el administrador está logueado en el sistema.

Este filtro se puede utilizar al tener un administrador logueado en el sistema. Se deberá usar al permitir a un conjunto de usuarios (ya sea administrador o otros roles que puedan ser agregados) acceder a ciertas funcionalidades del sistema.

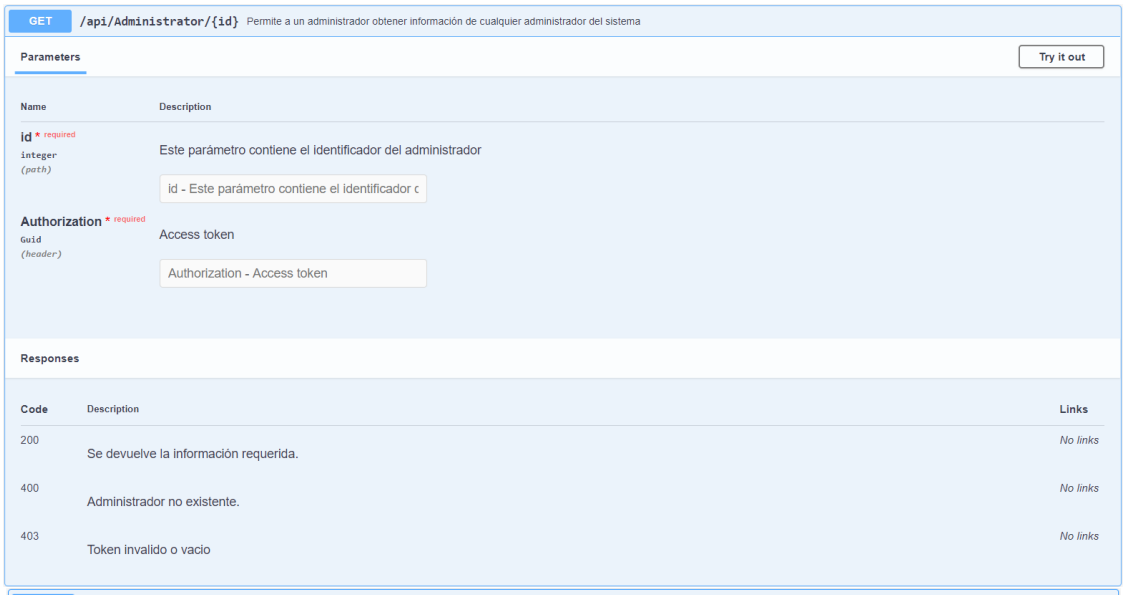
3. Códigos de error utilizados

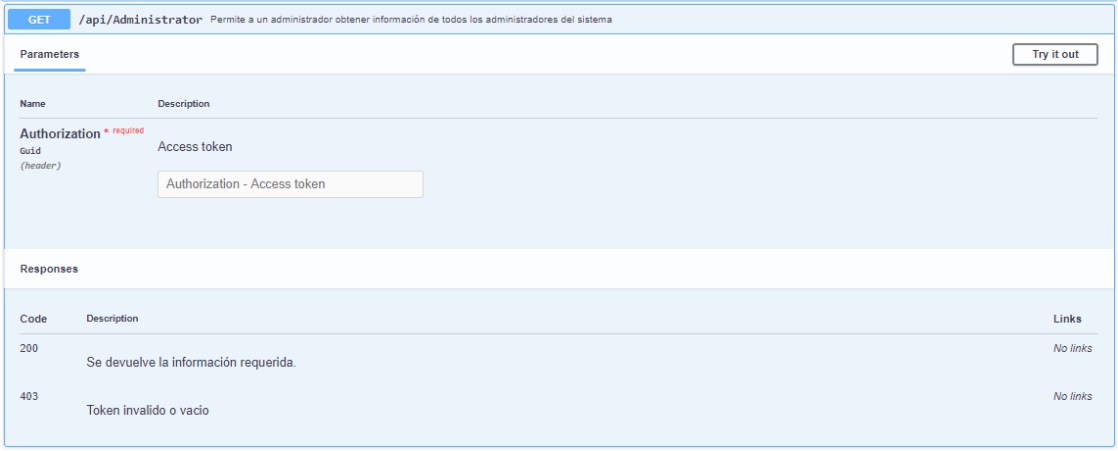
- OK Status code : 200 Indica que la acción realizada se ejecutó correctamente
- Bad Request Status Code : 400 Indica que hubo un error al realizar la acción requerida.
- Forbidden Status Code : 403 Indica que el servidor se niega a permitir la acción requerida.

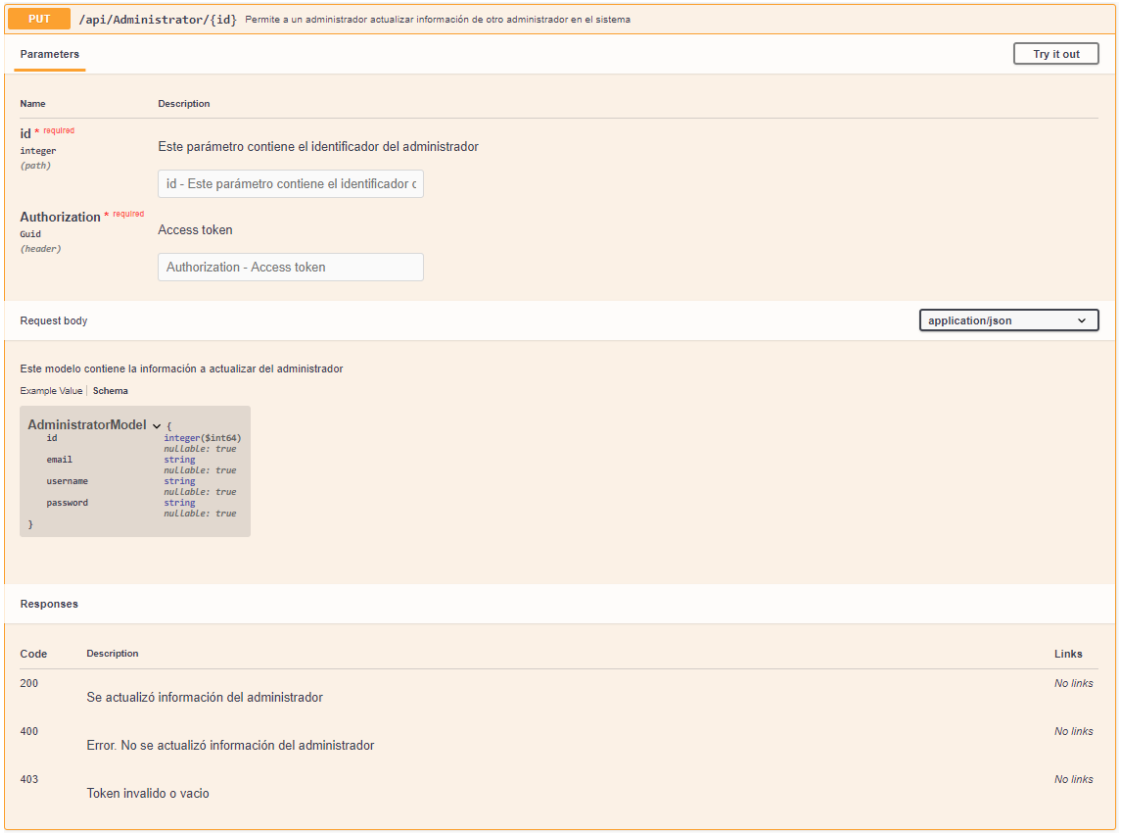
4. Especificación de endpoints

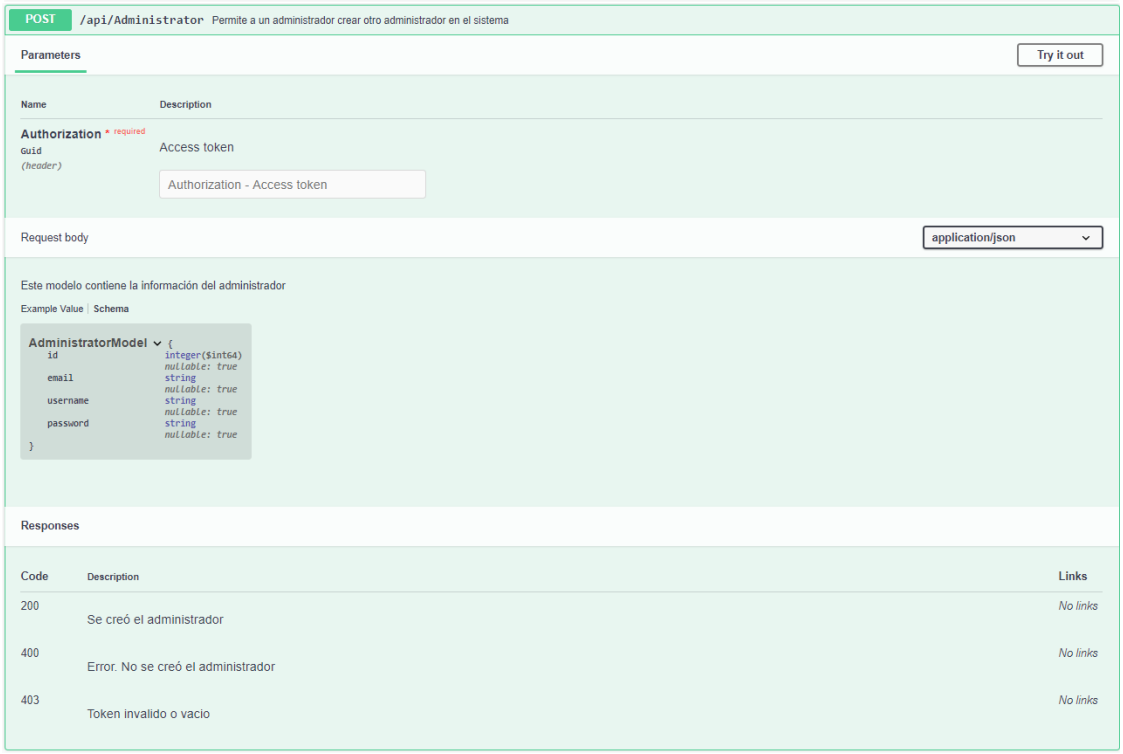
Para realizar esta parte, se utilizó el paquete de Swagger [3] para .Net Core.

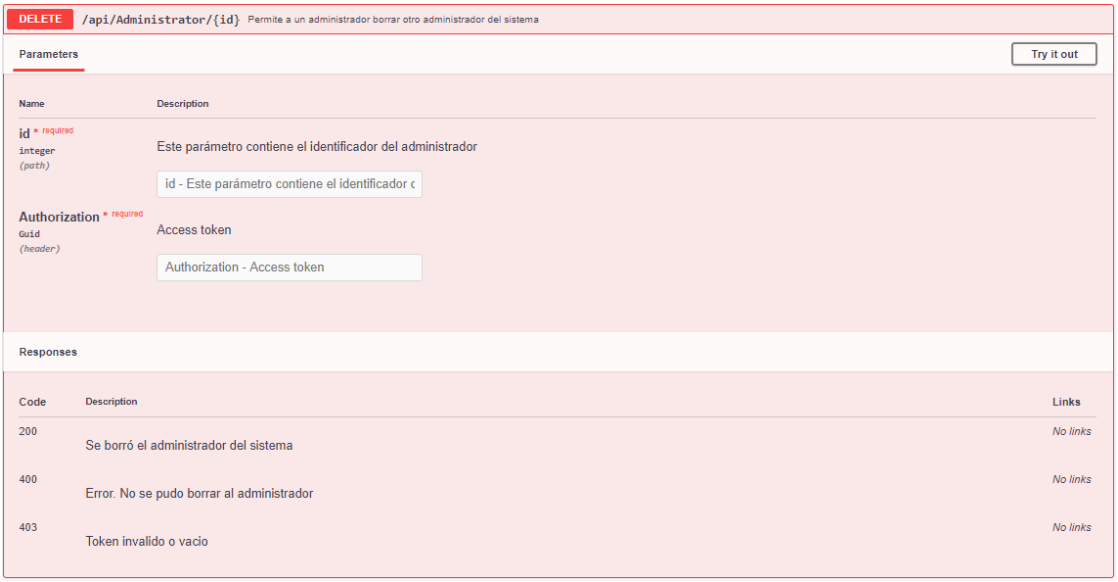
4.0.1. Administrator Controller



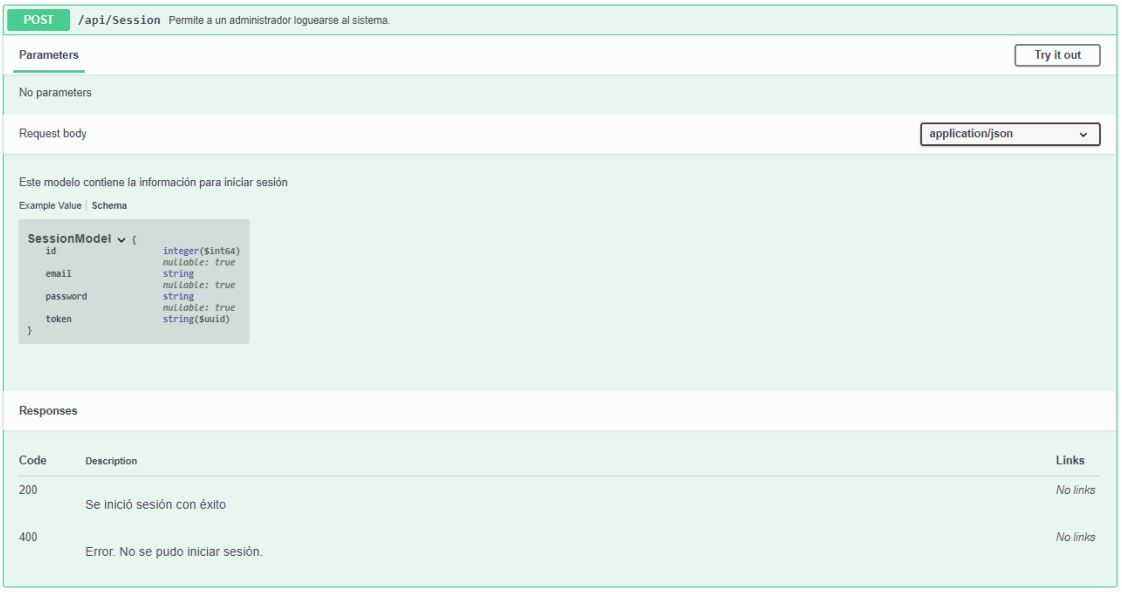



PUT
/api/Administrator/{id}
Permite a un administrador actualizar información de otro administrador en el sistema
Parameters
Try it out
Name
Description
id * required
integer
(path)
Este parámetro contiene el identificador del administrador
id - Este parámetro contiene el identificador c
Authorization * required
Guid
(header)
Access token
Authorization - Access token
Request body
application/json
Este modelo contiene la información a actualizar del administrador
Example Value | Schema
AdministratorModel
{
id integer($int64) nullable: true
email string nullable: true
username string nullable: true
password string nullable: true
}
Responses
Code
Description
Links
200
Se actualizó información del administrador
No links
400
Error. No se actualizó información del administrador
No links
403
Token invalido o vacio
No links

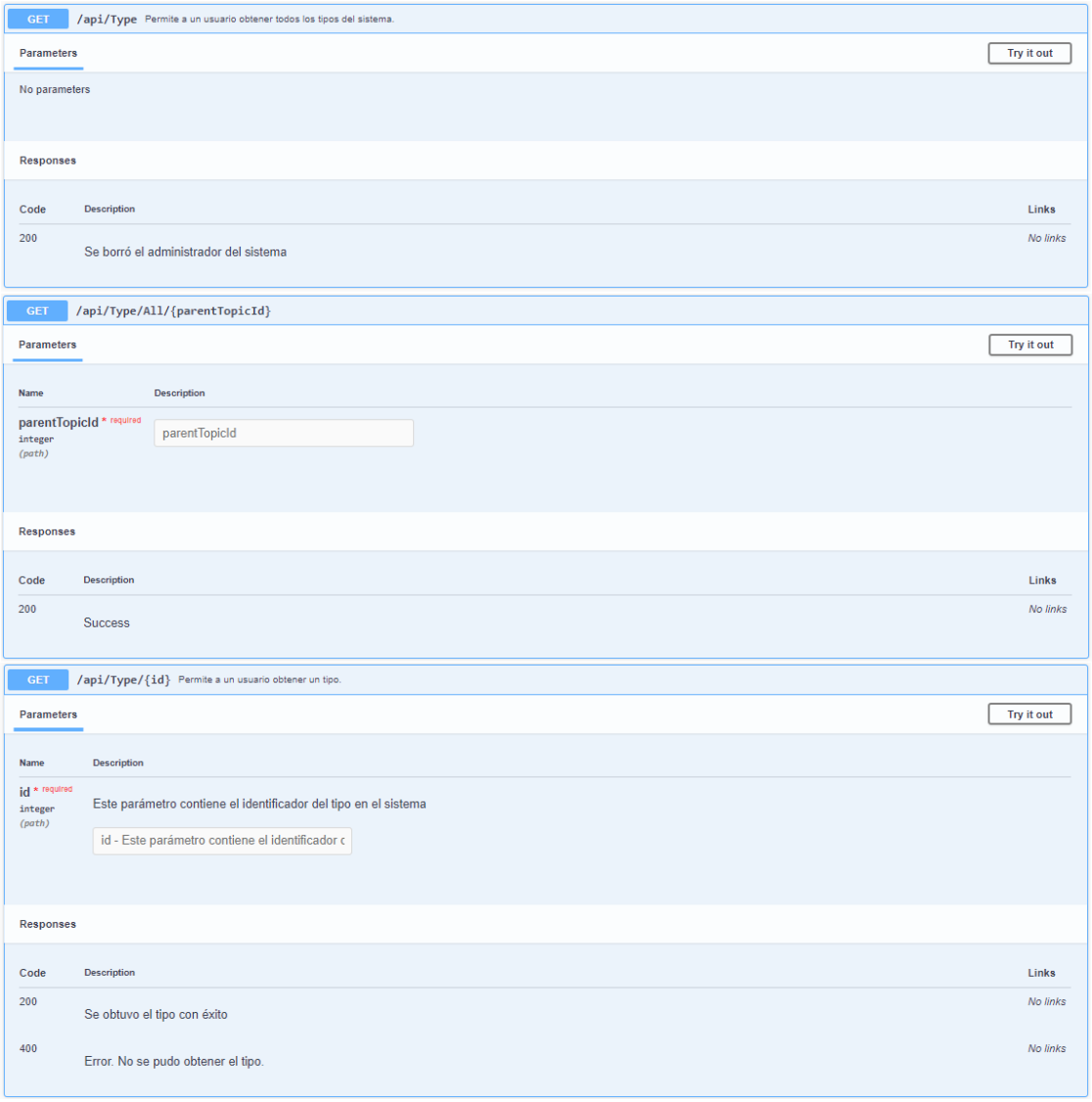

POST
/api/Administrator
Permite a un administrador crear otro administrador en el sistema
Parameters
Try it out
Name
Description
Authorization * required
Guid
(header)
Access token
Authorization - Access token
Request body
application/json
Este modelo contiene la información del administrador
Example Value | Schema
AdministratorModel
{
id integer($int64) nullable: true
email string nullable: true
username string nullable: true
password string nullable: true
}
Responses
Code
Description
Links
200
Se creó el administrador
No links
400
Error. No se creó el administrador
No links
403
Token invalido o vacio
No links

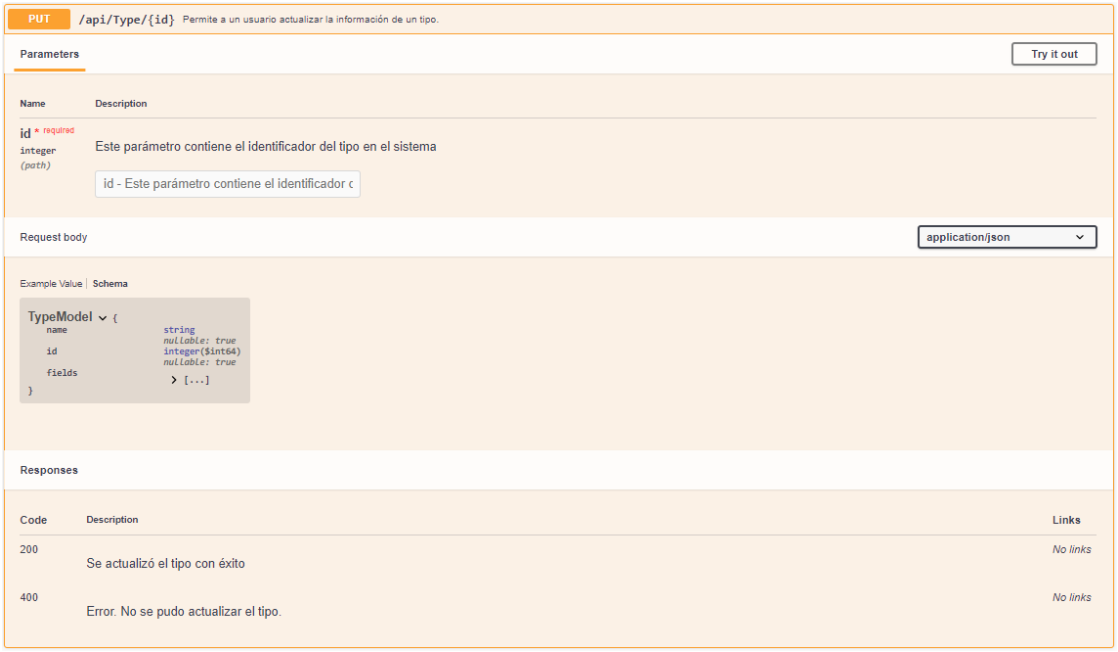


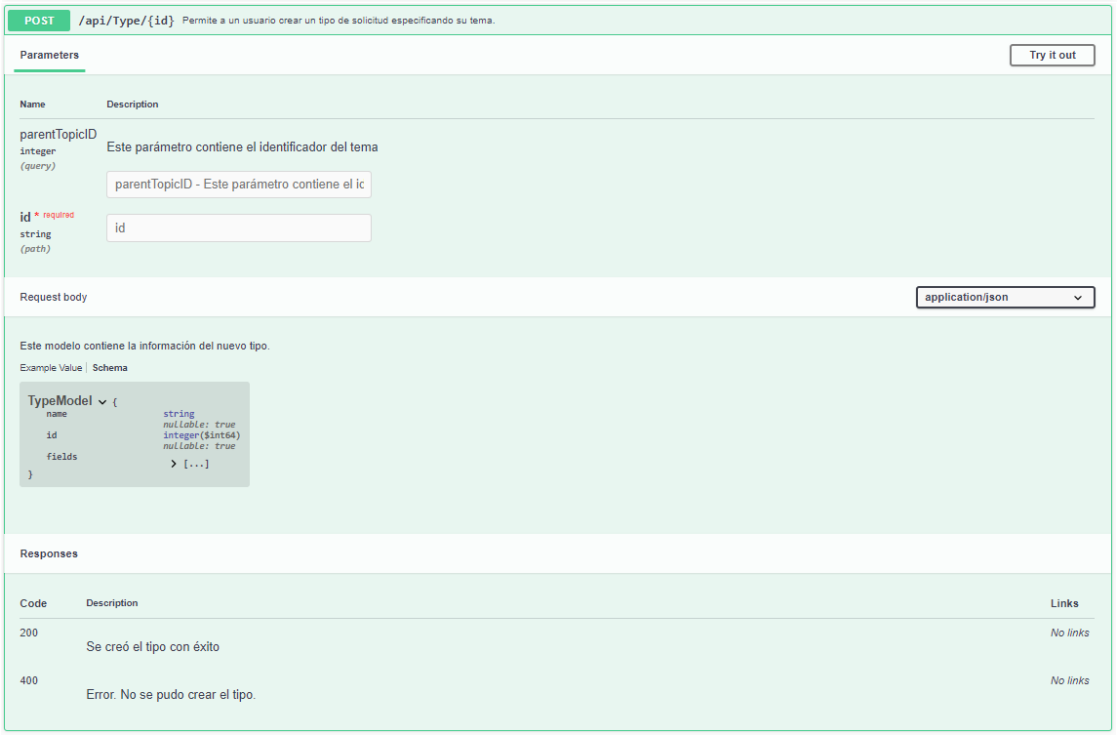
4.0.2. Session Controller

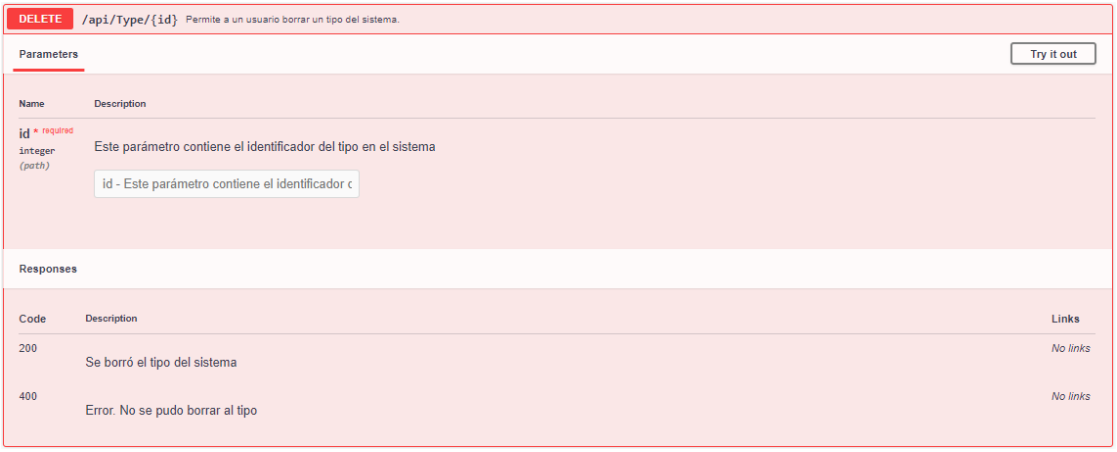


4.0.3. Type Controller




PUT
/api/Type/{id}
Permite a un usuario actualizar la información de un tipo.
Parameters
Try it out
Name
Description
id * required
integer
(path)
Este parámetro contiene el identificador del tipo en el sistema
id - Este parámetro contiene el identificador c
Request body
application/json
Example Value | Schema
TypeModel
name
string
nullable: true
id
integer($int64)
nullable: true
fields
[...]
Responses
Code
Description
Links
200
Se actualizó el tipo con éxito
No links
400
Error. No se pudo actualizar el tipo.
No links


POST
/api/Type/{id}
Permite a un usuario crear un tipo de solicitud especificando su tema.
Parameters
Try it out
Name
Description
parentTopicID
integer
(query)
Este parámetro contiene el identificador del tema
parentTopicID - Este parámetro contiene el ic
id * required
string
(path)
id
Request body
application/json
Este modelo contiene la información del nuevo tipo.
Example Value | Schema
TypeModel
name
string
nullable: true
id
integer($int64)
nullable: true
fields
[...]
Responses
Code
Description
Links
200
Se creó el tipo con éxito
No links
400
Error. No se pudo crear el tipo.
No links


DELETE
/api/Type/{id}
Permite a un usuario borrar un tipo del sistema.
Parameters
Try it out
Name
Description
id * required
integer
(path)
Este parámetro contiene el identificador del tipo en el sistema
id - Este parámetro contiene el identificador c
Responses
Code
Description
Links
200
Se borró el tipo del sistema
No links
400
Error. No se pudo borrar al tipo
No links

Bibliografía

[1] Universidad ORT Uruguay. (2013) Documento 302 - Facultad de Ingeniería. [Online]. Available: http://www.ort.edu.uy/fi/pdf/documento302facultaddeingenieria.pdf

[2] Microsoft Azure. (2018) API design. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

[3] Richard Morris. (2020) Swagger tools for documenting API's built on ASP.NET Core. [Online]. Available: https://github.com/domaindrivendev/Swashbuckle.AspNetCore

A. Visualizar documentación de la API online

En la carpeta *Documentación* se adjunta el archivo *swagger.json*. Este permite visualizar la documentación de la Api desde la página *https://editor.swagger.io/* de la siguiente forma:

- En el menú, ir a *File* -▷ *ImportFile*.
- Importar el archivo *swagger.json* encontrado en la carpeta *Documentación*.