

# Universidad ORT Uruguay

## **Obligatorio 1**

### **Diseño de Aplicaciones 2**

Agustina Disiot 221025

Ivan Monjardin 239850

<https://github.com/ORT-DA2/Disiot-221025-Monjardin-239850>

## *Índice:*

<b>Endpoints</b>	<b>3</b>
Admin:	3
Bug:	4
Developer:	7
Project:	8
Tester:	8

Para diseñar la api se diseñaron endpoint que cumplan con las recomendaciones de nombre de Rest Api, por ejemplo lo que estan detallado aca: <https://restfulapi.net/resource-naming/>

Algunas decisiones que se tomaron:

- Se definieron recursos como admin, project, developer y se escriben en plural al principio del endpoint
- Se utilizaron los métodos HTTP para indicar acciones sobre los recursos y no se incluyeron verbos en la URI. Por ejemplo:
  - GET /Bugs/{bug\_id} devuelve un bug específico
  - DELETE /Bugs/{bug\_id} lo elimina
- Todos los endpoints son en minúscula y no se incluye "/" al final
- Se utiliza tanto los parámetros por el body, los header o la route de la request dependiendo de que es mejor para cada caso


URL base: <http://localhost:5000/>

Códigos de error:


Como se mencionó en otro documento se establecieron códigos de error como 404 y 500 para cuando ocurre una falla en la petición del recurso. En caso que no ocurra error se devuelve 200 OK, al menos que sea una eliminación donde se puede devolver 201 NoContent

# Endpoints


## Admin:

**Admin** 

**POST** /admins

**Parameters** 

No parameters

Request body 

**Example Value** | **Schema**

```
{
  "id": 0,
  "username": "string",
  "name": "string",
  "lastname": "string",
  "password": "string",
  "email": "string"
}
```

Ejemplo de la request en el body:

```
{
  "username": "Administrator1",
  "name": "Lucía",
  "lastname": "López",
  "password": "admin123",
  "email": "lucia123@gmail.com"
}
```

## Bug:

Bug		▼
GET	/bugs	
POST	/bugs	
GET	/bugs/{id}	
PUT	/bugs/{id}	
DELETE	/bugs/{id}	
POST	/bugs/import/{format}	

Se definieron todas las operaciones para el Bug, y todas parten desde la misma URI “bugs”

Bug		▼
GET	/bugs	
Parameters		Try it out
No parameters		
Responses		
Code	Description	Links
200	Success	No links

Se devuelve 200 OK cuando se logra conseguir los bugs correctamente

POST	/bugs	
Parameters		Try it out
No parameters		
Request body		application/json ▼
Example Value Schema		

## Ejemplo:

```
{
  "name": "Error en el envío de correo",
  "description": "El error se produce cuando el usuario no tiene un correo asignado",
  "version": "1.0",
  "ProjectId": 2
}
```

El Post, tanto en bugs, como en los otros recursos, se definió que tiene que mandar la información por el Body. Así como son consistentes entre todos los recursos

En caso que se quiere realizar acciones sobre un bug en particular no es necesario ir a la ruta del proyecto, se aprovechó la ruta ya existente de bugs para simplificar la URI y solo tener que pasar la información importante. La misma URI sirve para modificar, conseguir o eliminar el bug, dependiendo del verbo HTTP

GET

/bugs/{id}

Parameters

Name	Description
<b>id</b> <small>★ required</small> <small>integer(\$int32)</small> <small>(path)</small>	<input type="text" value="id"/>

Responses

Code	Description
200	Success

PUT

/bugs/{id}

Parameters

Try it out

Name	Description
<b>id</b> <small>★ required</small> <small>integer(\$int32)</small> <small>(path)</small>	<input type="text" value="id"/>

Request body

application/json

Format: Media / Schema

**DELETE**
/bugs/{id}

### Parameters

Name	Description
<b>id</b> * required integer(\$int32) (path)	<input type="text" value="id"/>

### Responses

Code	Description
200	Success

Para la importación de bugs se definió que se elija el formato o “empresa” en la URI y el path al archivo en el servidor que esté en los headers. De esta manera, no hay complicaciones a la hora de probar el path en Postman u otra herramienta. Si se pusiera en la URI mismo los espacios y los “/” podrían complicar la lectura.

Entendemos que una regla es no incluir formato de archivos en las URI, sin embargo en este caso, el formato representa más una empresa que el formato mismo. Un posible “formato” podría ser “empresa2” y no necesariamente cada formato estar asociado a un tipo de archivos en particular.

**POST**
/bugs/import/{format}

### Parameters

Try it out

Name	Description
path string (header)	<input type="text" value="path"/>
<b>format</b> * required string (path)	<input type="text" value="format"/>

### Responses

Code	Description	Links
200	Success	No links

## Developer:

Al igual que para bugs se definió una ruta donde devs sea el principal actor.

<b>POST</b>	/devs
<b>Parameters</b>	
No parameters	
Request body	

## Ejemplo:

```
{
  "username": "juanperez",
  "name": "Juan",
  "lastname": "Perez",
  "password": "123dr568",
  "email": "juanperez@gmail.com"
}
```

<b>GET</b>	/devs/{id}/bugs/quantity
<b>Parameters</b>	
<b>Name</b>	<b>Description</b>
<b>idDev</b> * required integer(\$int32) (path)	idDev



## Project:

Project		▼
GET	/projects	
POST	/projects	
GET	/projects/{id}	
PUT	/projects/{id}	
DELETE	/projects/{id}	
GET	/projects/name/{name}	
GET	/projects/{id}/bugs	
GET	/projects/{id}/bugs/quantity	
GET	/projects/{id}/devs	
GET	/projects/{id}/testers	
DELETE	/projects/{idProject}/devs/{idDev}	
POST	/projects/{idProject}/devs/{idDev}	
DELETE	/projects/{idProject}/testers/{idTester}	
POST	/projects/{idProject}/testers/{idTester}	

Las acciones que donde si era necesario especificar el proyecto, se decidió incluir en la ruta de projects.

## Tester:

Una de las ideas importantes que consideramos para la URI era que fuera consistente. Es decir, que no se tenga que usar un formato de URI para los devs y otro para los testers. Ejemplo, el POST es similar para ambos y sucede lo mismo con el proyecto.

Tester		▼
POST	/testers	
GET	/testers/{idTester}/bugs/status/{filter}	
GET	/testers/{idTester}/bugs/name/{filter}	
GET	/testers/{idTester}/bugs/project/{filter}	