

# APIs Para Naves

## Contenido

1. Crear Nave .....	2
1.1 Formato esperado del cuerpo de la petición .....	2
1.2 Respuesta esperada .....	2
2. Obtener lista de naves .....	3
2.1 Formato esperado del cuerpo de la petición .....	3
2.2 Resultado esperado .....	3
2.3 Consideraciones.....	5
3. Devolver nave por ID .....	5
3.1 Formato esperado del cuerpo de la petición .....	5
3.2 Resultado esperado .....	5
4. Devolver lista de nave vía un filtro por nombre .....	5
4.1 Formato esperado del cuerpo de la petición .....	5
4.2 Resultado esperado .....	6
5. Actualización de nave por nombre .....	7
5.1 Formato esperado del cuerpo de la petición .....	7
5.2 Resultado esperado .....	7
6. Actualización de nave por nombre .....	7
6.1 Formato esperado del cuerpo de la petición .....	7
6.2 Resultado esperado .....	7
7. API para mensajería .....	8
7.1 Formato esperado del cuerpo de la petición .....	8
7.2 Resultado esperado .....	8

# 1. Crear Nave

## 1.1 Formato esperado del cuerpo de la petición

La API encargada de esta funcionalidad es :

[localhost:8080/spaceships/create-spaceship](http://localhost:8080/spaceships/create-spaceship)

Esta API en un POST y se encarga de crear un nave, via el nombre.

El formato del cuerpo de la solicitud es:

```
{  
  "name": ""  
}
```

Un ejemplo es:

```
{  
  "name": "elton1"  
}
```

## 1.2 Respuesta esperada

Ejemplo de la respuesta esperada es:

```
{  
  "id": 1,
```

```
    "name": "elton1"  
  }
```

el estado de la respuesta será de 200 OK.

## 2. Obtener lista de naves

### 2.1 Formato esperado del cuerpo de la petición

La API encargada de esta funcionalidad es :

[Localhost:8080/spaceships/all-spaceships](http://localhost:8080/spaceships/all-spaceships)

Esta API es un POST y se encarga de generar la lista de naves completo.

El formato del cuerpo es:

```
{  
  "direction": "", "sortBy": "", "page":  
}
```

Un ejemplo es:

```
{  
  "direction": " ASC ", "sortBy": " id ", "page": 0  
}
```

Así, la «página» representa el número de página de la que queremos obtener elementos. si página es 0, devolvemos los 4 primeros elementos, y si es uno, devolvemos los 4 siguientes elementos en la página siguiente y así sucesivamente.

«sortBy» es el campo por el que queremos ordenar, y «direction», la dirección de la ordenación.

### 2.2 Resultado esperado

Por un ejemplo de request body:

```
{  
  "direction": " ASC ", "sortBy": " id ", "page": 0  
}
```

La respuesta esperada, por un lista de 6 elementos existentes en la base de datos es:

```
{  
  "content": [  
    {  
      "id": 1,  
      "name": "elton1"  
    },  
    {  
      "id": 2,  
      "name": "elton2-test"  
    },  
    {  
      "id": 3,  
      "name": "elton3-x"  
    },  
    {  
      "id": 4,  
      "name": "elton4-test"  
    }  
  ]  
}
```

```

    }
  ],
  "pageable": {
    "pageNumber": 0,
    "pageSize": 4,
    "sort": {
      "empty": false,
      "sorted": true,
      "unsorted": false
    },
    "offset": 0,
    "unpaged": false,
    "paged": true
  },
  "last": false,
  "totalPages": 2,
  "totalElements": 6,
  "size": 4,
  "number": 0,
  "sort": {
    "empty": false,
    "sorted": true,
    "unsorted": false
  },
  "first": true,
  "numberOfElements": 4,
  "empty": false
}

```

Para ver un ejemplo de un cuerpo de solicitud para la misma lista de elementos:

```

{
  "direction": " ASC ", "sortBy": " id ", "page": 1
}

```

La respuesta esperada es:

```

{
  "content": [
    {
      "id": 5,
      "name": "elton5"
    },
    {
      "id": 6,
      "name": "elton6"
    }
  ],
  "pageable": {
    "pageNumber": 1,
    "pageSize": 4,
    "sort": {
      "empty": false,
      "sorted": true,
      "unsorted": false
    },
    "offset": 0,
    "unpaged": false,
    "paged": true
  },
  "last": false,
  "totalPages": 2,
  "totalElements": 6,
  "size": 4,
  "number": 0,
}

```

```
"sort": {
  "empty": false,
  "sorted": true,
  "unsorted": false
},
"first": true,
"numberOfElements": 2,
"empty": false
}
```

el estado de la respuesta será de 200 OK.

## 2.3 Consideraciones

Como no había especificaciones sobre el número de elementos por página, lo mantuve en un máximo de 4 elementos por página, en el Paginated response Body.

# 3. Devolver nave por ID

## 3.1 Formato esperado del cuerpo de la petición

La API encargada de esta funcionalidad es :

[localhost:8080/spaceships/get-spaceship/{id}](http://localhost:8080/spaceships/get-spaceship/{id})

Esta API en un GET y se encarga de devolver un nave único por id.

No tiene cuerpo de petición.

## 3.2 Resultado esperado

Por un ejemplo de petición,

[localhost:8080/spaceships/get-spaceship/1](http://localhost:8080/spaceships/get-spaceship/1)

El resultado esperado es:

```
{
  "id": 1,
  "name": "elton1"
}
```

el estado de la respuesta será de 200 OK.

# 4. Devolver lista de nave vía un filtro por nombre

## 4.1 Formato esperado del cuerpo de la petición

La API encargada de esta funcionalidad es :

[localhost:8080/spaceships/filter-spaceships](http://localhost:8080/spaceships/filter-spaceships)

Esta API en un POST y devuelve una lista de naves espaciales con un nombre que contiene la cadena pasada en el cuerpo de la petición.

El cuerpo esperado es:

```
{
  "object": {
    "filter": ""
  },
  "direction": "", "sortBy": "", "page": 0
}
```

## 4.2 Resultado esperado

Por un ejemplo de petición,

```
{
  "object": {
    "filter": "test"
  },
  "direction": "ASC", "sortBy": "id", "page": 0
}
```

El resultado esperado es:

```
{
  "content": [
    {
      "id": 2,
      "name": "elton2-test"
    },
    {
      "id": 4,
      "name": "elton4-test"
    }
  ],
  "pageable": {
    "pageNumber": 0,
    "pageSize": 4,
    "sort": {
      "empty": false,
      "sorted": true,
      "unsorted": false
    },
    "offset": 0,
    "unpaged": false,
    "paged": true
  },
  "last": false,
  "totalPages": 1,
  "totalElements": 2,
  "size": 4,
  "number": 0,
  "sort": {
    "empty": false,
    "sorted": true,
    "unsorted": false
  },
  "first": true,
}
```

```
"numberOfElements": 2,  
"empty": false  
}
```

el estado de la respuesta será de 200 OK.

## 5. Actualización de nave por nombre

### 5.1 Formato esperado del cuerpo de la petición

La API encargada de esta funcionalidad es :

[Localhost:8080/spaceships/update-spaceship/{id}](http://localhost:8080/spaceships/update-spaceship/{id})

Esta API en un PUT y actualiza la nave con el id de la petición. El formato del cuerpo es:

```
{  
  "name": ""  
}
```

### 5.2 Resultado esperado

Es Por un ejemplo de cuerpo de petición:

```
{  
  "name": "elton"  
}
```

Con url: [Localhost:8080/spaceships/update-spaceship/1](http://localhost:8080/spaceships/update-spaceship/1)

La respuesta esperada es:

```
{  
  "id": 1,  
  "name": "elton"  
}
```

el estado de la respuesta será de 200 OK.

## 6. Actualización de nave por nombre

### 6.1 Formato esperado del cuerpo de la petición

La API encargada de esta funcionalidad es :

[Localhost:8080/spaceships/delete-spaceship/{id}](http://localhost:8080/spaceships/delete-spaceship/{id})

Esta API en un DELETE y se encarga de borrar un nave por id.

### 6.2 Resultado esperado

Por un ejemplo de petición,

[Localhost:8080/spaceships/get-spaceship/1](http://localhost:8080/spaceships/get-spaceship/1)

No hay un resultado esperado. Pero el estado de la respuesta será de 404 NOT FOUND.

## 7. API para mensajería

### 7.1 Formato esperado del cuerpo de la petición

La API se encargada de un sistema de mensajería de Kafka. El API de esta funcionalidad es :

[localhost:8080/spaceships/message-spaceships](http://localhost:8080/spaceships/message-spaceships)

### 7.2 Resultado esperado

No hay cuerpo de recepción.

Solo el estado de la respuesta será de 200 OK.