



Universidad ORT Uruguay

Facultad de Ingeniería

Documentación Obligatorio 1

Evidencia de la ejecución de las pruebas de la API con Postman

Diseño de Aplicaciones 2

Link al Repositorio: [IngSoft-DA2/301178_231810_280070 \(github.com\)](https://github.com/IngSoft-DA2/301178_231810_280070)

Integrantes:

- Angelina Maverino - 280070
- Yliana Otero - 301178
- María Belén Rywaczuk - 231810

Tutores:

- Juan Ignacio Irabedra De Maio
- Juan Pablo Barrios García
- Ignacio Valle Dubé

Índice

Índice.....	2
Demo SmartHome API con Postman.....	3
Links.....	3
Video de YouTube.....	3
Colección de Postman.....	3
Script para crear datos de prueba.....	3
Documento de evidencia de diseño y especificación de la API.....	3
Reporte.....	4
Mantenimiento de cuentas de administrador.....	4
Creación de cuenta de administrador.....	4
Camino correcto.....	4
Administrador ya existe.....	5
Propiedades del usuario no cumplen con los requisitos.....	5
Faltan propiedades en el cuerpo de la solicitud.....	6
El cuerpo de la solicitud está vacío.....	7
No se incluye Authorization header.....	7
Usuario no tiene permisos.....	8
Token invalido.....	8
Token no corresponde a ningún usuario.....	9
Eliminación de cuenta de administrador.....	9
Camino correcto.....	9
No se encuentra administrador con el ID.....	10
Creación de una empresa.....	10
Camino correcto.....	10
El ownerId no corresponde a un CompanyOwner sin empresa asociada.....	11
Detección de movimiento.....	12
Camino correcto.....	12
Asociar dispositivos al hogar.....	12
Camino correcto.....	13
No se encuentra el dispositivo.....	13
No se encuentra el hogar.....	14

Demo SmartHome API con Postman

Se realizó una demostración del uso de la SmartHome API con Postman. La demo puede verse desde YouTube. En la demo se muestran (y comentan) varios escenarios posibles para todos los endpoints desarrollados, los cuales se encuentran en la colección de Postman. La colección también tiene una descripción de cada uno de los endpoints.

Links

Video de YouTube

https://www.youtube.com/watch?v=Hv_-NmVIOEw

Colección de Postman

Drive:

https://drive.google.com/file/d/1Qejco0Jh0EEfxWMFnx_y3jjvqEW_nXG5p/view?usp=sharing

GitHub:

https://github.com/IngSoft-DA2/301178_231810_280070/blob/main/Documentacion/SmartHome%20API.postman_collection.json

Script para crear datos de prueba

Drive:

<https://drive.google.com/file/d/1PopwrZo1DgX46TBvPM13IbUDby5ONWA-/view?usp=sharing>

GitHub:

https://github.com/IngSoft-DA2/301178_231810_280070/blob/main/Base%20de%20Datos/insert_test_data.sql

Documento de evidencia de diseño y especificación de la API

Drive:

https://drive.google.com/file/d/1IyoBypeCaZg_349X4NgT4WXSxIX8boYe/view?usp=sharing

GitHub:

https://github.com/IngSoft-DA2/301178_231810_280070/blob/main/Documentacion/Evidencia%20del%20dise%C3%B1o%20y%20especificaci%C3%B3n%20de%20la%20API.pdf

Reporte

Mantenimiento de cuentas de administrador

Creación de cuenta de administrador:

Camino correcto

Ruta:

POST /api/v1/administrators

Request headers

Authorization	[[token GUID]]
---------------	----------------

El token GUID es un token que cumple:

- No está vacío.
- Es un formato GUID válido.
- Corresponde a la sesión de un usuario.
- El usuario tiene el rol Administrator.

Request body:

No es vacío y tiene los siguientes campos:

- Name (string no vacío) – el nombre del administrador a crear.
- Surname (string no vacío) – el apellido del administrador a crear.
- Email (string no vacío que cumple el regex
"^[a-zA-Z0-9_+-.]+@[a-zA-Z0-9-]+\.[a-zA-Z0-9-]+\$") y no existe ya un usuario creado con
ese email – el correo electrónico del administrador a crear.
- Contraseña (string no vacío que cumple el regex
"^(?=.*[#@\$,%])(?=.*[a-z])(?=.*[A-Z])(?=.*\d){8,}\$") – la contraseña del administrador a
crear.

Response code:

201 - Created

Response body:

Modelo de respuesta para POST administrators (nombre, apellido y email del administrador creado).

Evidencia:

POST {{host}} {{base_route}}/administrators

Params Authorization Headers (10) Body • Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▾

```
1 {
2   "name": "Jackson",
3   "surname": "Belleville",
4   "email": "jacksonbelleville@gmail.com",
5   "photo": "https://static.wikia.nocookie.net/gilmoregirls/images/d/d7/Jackson_zpsbc778b3a.jpg/revision/latest?cb=20131229072506",
6   "password": "Password1%"
7 }
```

Body Cookies Headers (5) Test Results 201 Created

Pretty Raw Preview Visualize JSON ▾

```
1 {
2   "name": "Jackson",
3   "email": "jacksonbelleville@gmail.com",
4   "surname": "Belleville"
5 }
```

Administrador ya existe

Ruta OK

Request headers OK

Request body:

No está vacío y tiene todos los campos mencionados en el camino correcto, pero ya existe un usuario creado con el email.

Response code:

409 - Conflict

Response body:

Mensaje “Member already exists”.

Evidencia:

POST {{host}} {{base_route}}/administrators

Params Authorization Headers (10) Body • Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL JSON ▾

```
1 {
2   "name": "Jackson",
3   "surname": "Belleville",
4   "email": "jacksonbelleville@gmail.com",
5   "photo": "https://static.wikia.nocookie.net/gilmoregirls/images/d/d7/Jackson_zpsbc778b3a.jpg/revision/latest?cb=20131229072506",
6   "password": "Password1%"
7 }
```

Body Cookies Headers (4) Test Results 409 Conflict

Pretty Raw Preview Visualize JSON ▾

```
1 {
2   "message": "Member already exists"
3 }
```

Propiedades del usuario no cumplen con los requisitos

Ruta OK

Request headers OK

Request body:

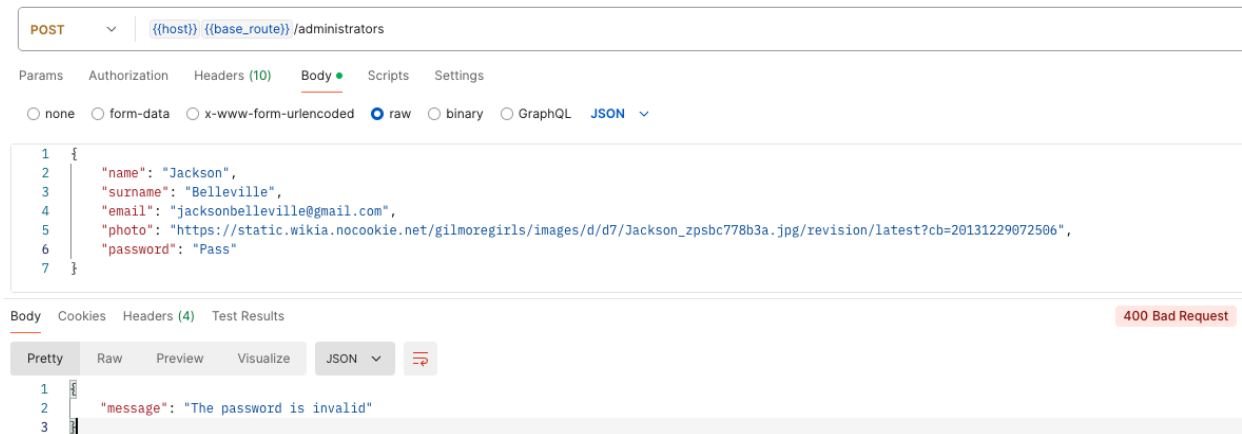
No está vacío y tiene todos los campos mencionados en el camino correcto, pero alguno de ellos no cumple con los requisitos. Por ejemplo, la contraseña no cumple con el regex.

Response code:

400 - Bad Request

Response body:

Mensaje “The password is invalid”.

Evidencia:

Faltan propiedades en el cuerpo de la solicitud

Ruta OK**Request headers OK****Request body:**

Faltan campos requeridos (por ejemplo el body es {}).

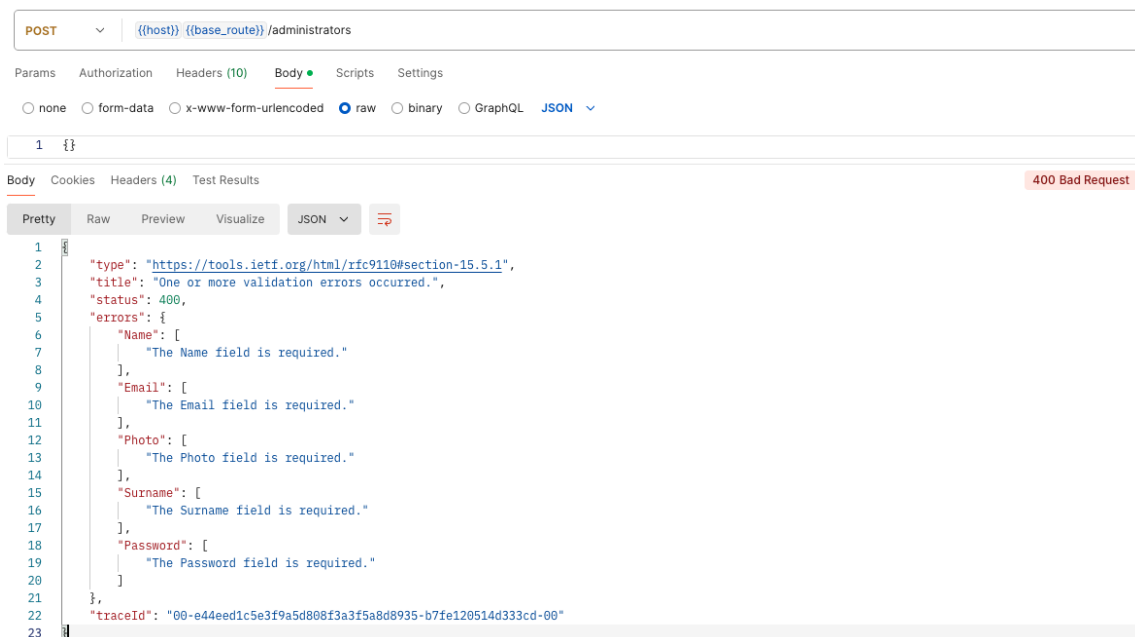
Response code:

400 - Bad Request

Response body:

Respuesta estándar de campos requeridos.

Evidencia:



El cuerpo de la solicitud está vacío

Ruta OK

Request headers OK

Request body:

Está vacío.

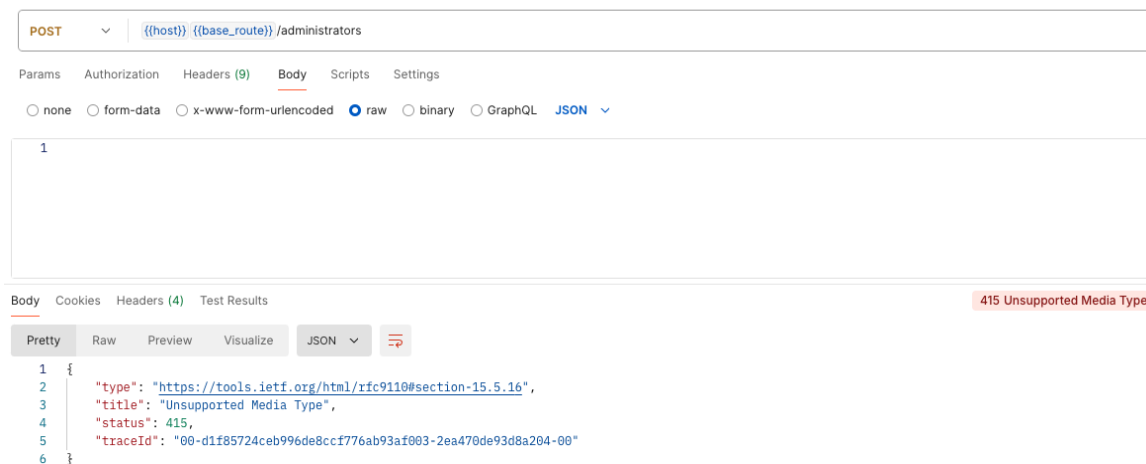
Response code:

415 - Unsupported Media Type.

Response body:

Modelo de respuesta estándar de Unsupported Media Type.

Evidencia:



No se incluye Authorization header

Request headers:

Falta Authorization header

Response code:

401 - Unauthorized

Response body:

“Authorization header is needed”

Evidencia:

The screenshot shows a REST client interface for a POST request to `{{(host)}} {{(base_route)}} /administrators`. The 'Headers' tab is active, showing a table with headers: 'Connection' (keep-alive) and 'Authorization' ({{(administratorToken)}}). The 'Body' tab is also visible, showing a message: '1 Authorization header is needed'. The response status is '401 Unauthorized'.

Key	Value	Description
Connection	keep-alive	
Authorization	{{(administratorToken)}}	

1 Authorization header is needed

401 Unauthorized

Usuario no tiene permisos

Request headers:

Se incluye Authorization header pero el usuario correspondiente al token no tiene permisos suficientes (no es Administrator).

Response code:

403 - Forbidden

Response body:

“The user does not have the necessary permissions”

Evidencia:

The screenshot shows a REST client interface for a POST request to `{{(host)}} {{(base_route)}} /administrators`. The 'Headers' tab is active, showing a table with headers: 'Connection' (keep-alive) and 'Authorization' ({{(homeOwnerRichardToken)}}). The 'Body' tab is also visible, showing a message: '1 The user does not have the necessary permissions'. The response status is '403 Forbidden'.

Key	Value	Description
Connection	keep-alive	
Authorization	{{(homeOwnerRichardToken)}}	

1 The user does not have the necessary permissions

403 Forbidden

Token invalido

Request headers:

Se incluye Authorization header pero el token es inválido.

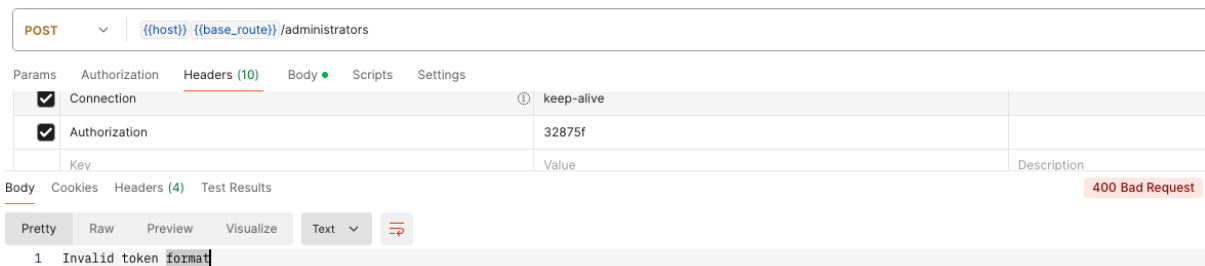
Response code:

400 - Bad Request

Response body:

“Invalid token format”

Evidencia:



Token no corresponde a ningún usuario

Request headers:

Se incluye Authorization header pero el token no corresponde a ningún usuario.

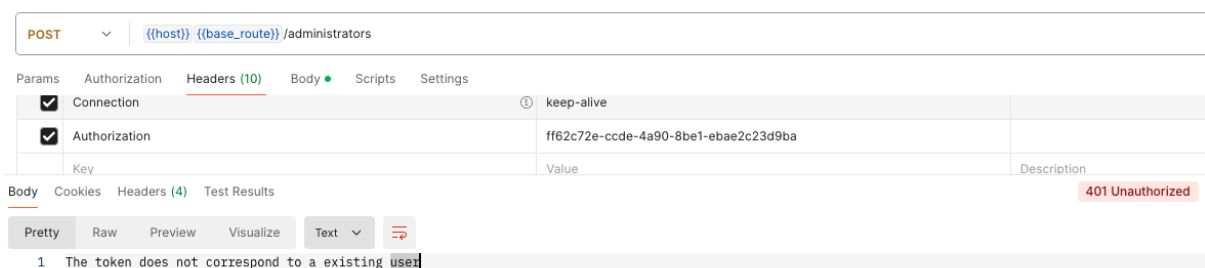
Response code:

401 - Unauthorized

Response body:

“The token does not correspond to a existing user”

Evidencia:



Eliminación de cuenta de administrador

Se requiere el mismo Authorization header con los mismos requisitos. Los casos “Usuario no tiene permisos”, “Token invalido”, “Token no corresponde a ningún usuario” y “No se incluye Authorization header” se comportan exactamente de la misma manera y son manejados por el AuthorizationFilter (ver documentación de evidencia de diseño y especificación de la API), por lo que no se vuelven a mostrar aquí.

Camino correcto

Ruta:

DELETE `/api/v1/administrators/{id}`

Donde `{id}` es el ID del usuario a eliminar, y este usuario existe.

Request headers OK

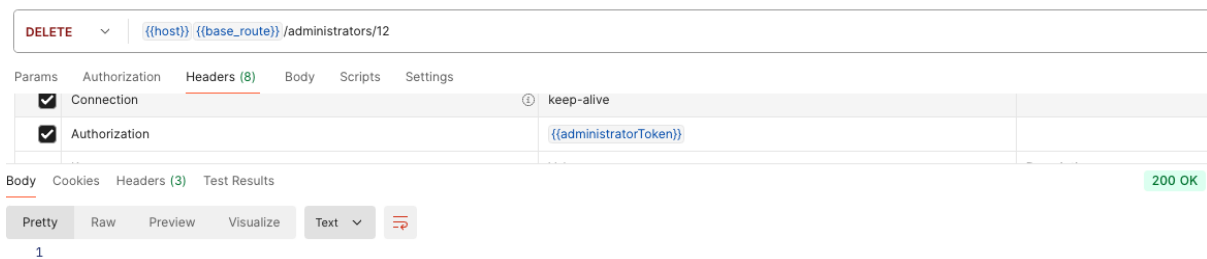
Response code:

200 - Ok

Response body:

Vacío.

Evidencia:



No se encuentra administrador con el ID

Ruta:

DELETE /api/v1/administrators/{id}

Donde {id} es el ID del usuario a eliminar, y este usuario no existe.

Request headers OK

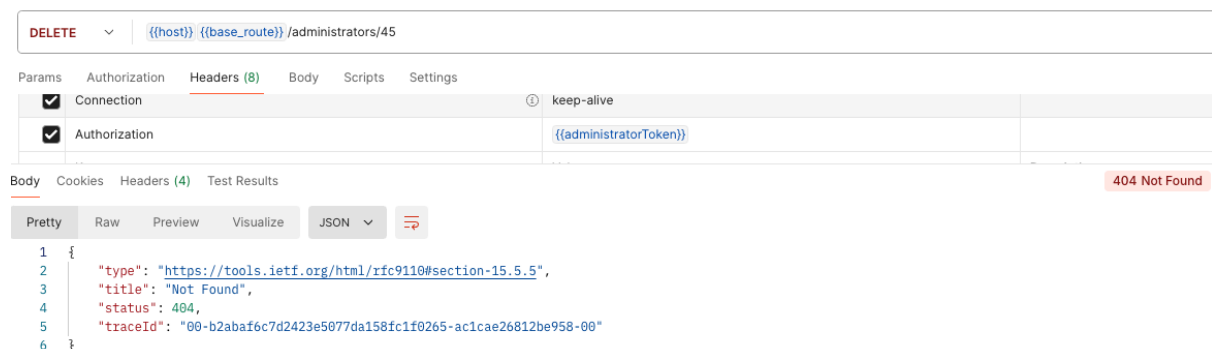
Response code:

404 - Not Found

Response body:

Respuesta estándar de Not Found.

Evidencia:



Creación de una empresa

Se requiere el mismo Authorization header con los mismos requisitos, salvo que en este caso el usuario debe tener rol de CompanyOwner. Los casos “Usuario no tiene permisos”, “Token invalido”, “Token no corresponde a ningún usuario” y “No se incluye Authorization header” se comportan exactamente de la misma manera y son manejados por el AuthorizationFilter (ver documentación de evidencia de diseño y especificación de la API), por lo que no se vuelven a mostrar aquí.

De la misma forma, no se muestran los casos “Faltan propiedades en el cuerpo de la solicitud” o “El cuerpo de la solicitud está vacío”.

Camino correcto

Ruta

POST /api/v1/companies

Request headers OK

Request body:

Tiene los campos “name” (string), “rut” (string que representa un long), “logoUrl” (string) y “ownerId” (long). El ownerId corresponde al Id de un usuario existente con el rol de CompanyOwner, que no tiene ninguna empresa asociada aun.

Response code:

201 Created

Response body:

Campos incluidos en el request body.

Evidencia:

The screenshot shows a REST client interface with a POST request to `{{(host)}} {{(base_route)}}/companies`. The request body is a JSON object with the following fields: `"name": "IoT Home", "rut": "473457534", "logoUrl": "https://t4.ftcdn.net/jpg/03/85/50/33/360_F_385503388_tBGecrjitRyBXbbDQVSH14BXacRvo0cX.jpg", "ownerId": 13`. The response status is 201 Created. The response body is displayed in JSON format, showing the same fields as the request body.

El ownerId no corresponde a un CompanyOwner sin empresa asociada

Ruta OK

Request headers OK

Request body:

El ownerId no corresponde al Id de un usuario existente con el rol de CompanyOwner que no tenga ninguna empresa asociada aun.

Response code:

404 Not Found

Response body:

“CompanyOwner was not found.”

Evidencia:

The screenshot shows a REST client interface with a POST request to `{{(host)}} {{(base_route)}}/companies`. The request body is a JSON object with the following fields: `"name": "IoT Home", "rut": "473457534", "logoUrl": "https://t4.ftcdn.net/jpg/03/85/50/33/360_F_385503388_tBGecrjitRyBXbbDQVSH14BXacRvo0cX.jpg", "ownerId": 5`. The response status is 404 Not Found. The response body is displayed in Text format, showing the message: `CompanyOwner was not found.`

Detección de movimiento

Se requiere el mismo Authorization header con los mismos requisitos, salvo que en este caso cualquier usuario autenticado funciona. Los casos “Token invalido”, “Token no corresponde a ningún usuario” y “No se incluye Authorization header” se comportan exactamente de la misma manera y son manejados por el AuthorizationFilter (ver documentación de evidencia de diseño y especificación de la API), por lo que no se vuelven a mostrar aquí.

De la misma forma, no se muestran los casos “Faltan propiedades en el cuerpo de la solicitud” o “El cuerpo de la solicitud está vacío”.

Camino correcto

Ruta

POST /api/v1/notifications

Request headers OK

Request body:

Tiene los campos “event” (string), “homeId” (long), y “hardwareId” (guid). El homeId corresponde a un hogar existente, y el hardwareId a un dispositivo existente en ese hogar.

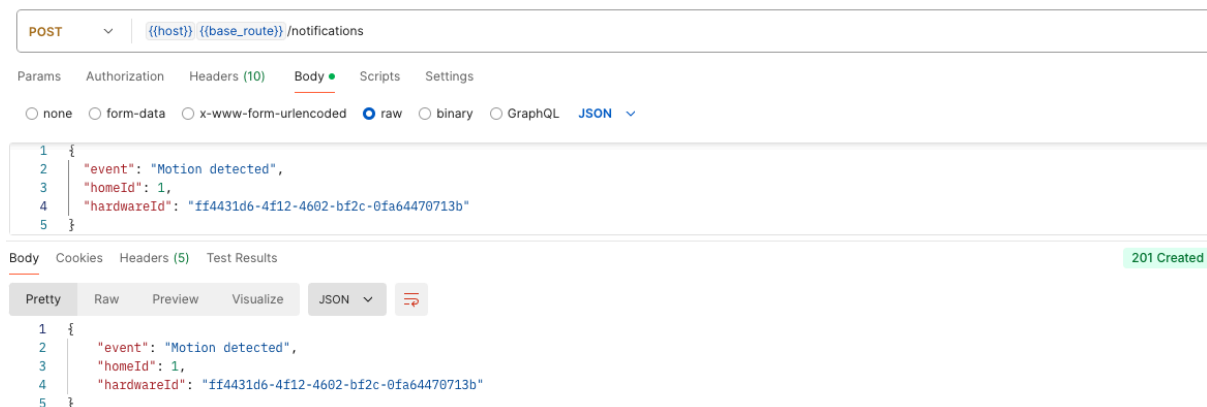
Response code:

201 Created

Response body:

Campos incluidos en el request body.

Evidencia:



Observación: si el dispositivo no está conectado o ninguno de los miembros del hogar recibe notificaciones, las instancias de notificaciones no se guardan en la base de datos.

Asociar dispositivos al hogar

Se requiere el mismo Authorization header con los mismos requisitos, salvo que en este caso el usuario debe ser el owner del hogar, o ser un miembro con permiso para asociar dispositivos al hogar. Los casos “Usuario no tiene permisos”, “Token invalido”, “Token no corresponde a ningún usuario” y “No se incluye Authorization header” se comportan exactamente de la misma manera y son manejados por el AuthorizationFilter (ver documentación de evidencia de diseño y especificación de la API), por lo que no se vuelven a mostrar aquí.

De la misma forma, no se muestran los casos “Faltan propiedades en el cuerpo de la solicitud” o “El cuerpo de la solicitud está vacío”.

Camino correcto

Ruta

POST /api/v1/home/{id}/devices

Donde {id} es el Id de un hogar existente en la plataforma.

Request headers OK

Request body:

Tiene el campo “deviceUnits” con una lista donde cada elemento contiene los campos “deviceId” (long) e “isConnected” (bool). El deviceId corresponde al Id de un dispositivo existente.

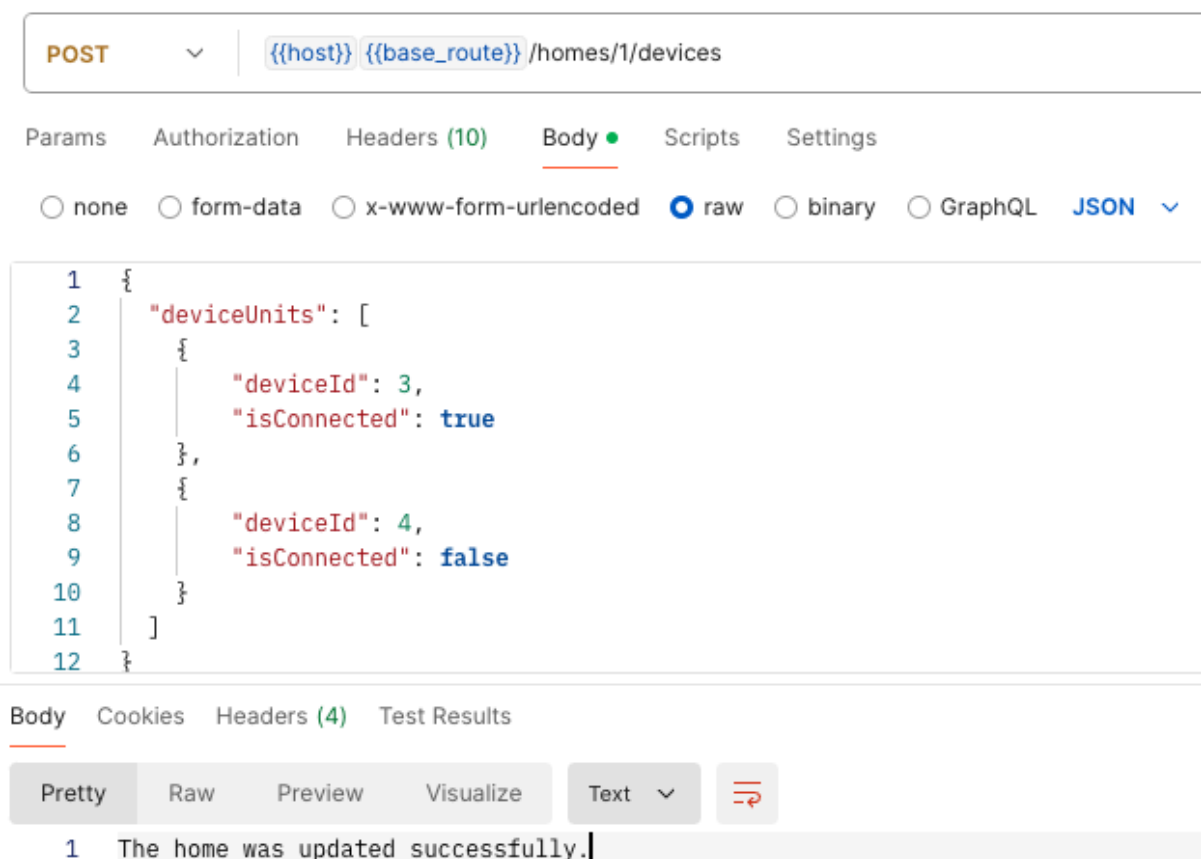
Response code:

200 OK

Response body:

“The home was updated successfully.”

Evidencia:



The screenshot displays a REST client interface with the following details:

- Method:** POST
- URL:** `{{host}} {{base_route}} /homes/1/devices`
- Body Type:** raw (selected), JSON
- Request Body (JSON):**

```
1 {
2   "deviceUnits": [
3     {
4       "deviceId": 3,
5       "isConnected": true
6     },
7     {
8       "deviceId": 4,
9       "isConnected": false
10    }
11  ]
12 }
```
- Response Body:** The home was updated successfully.

No se encuentra el dispositivo

Ruta OK

Request headers OK

Request body:

Tiene el campo “deviceUnits” con una lista donde cada elemento contiene los campos “deviceId” (long) e “isConnected” (bool). El deviceId corresponde al Id de un dispositivo no existente.

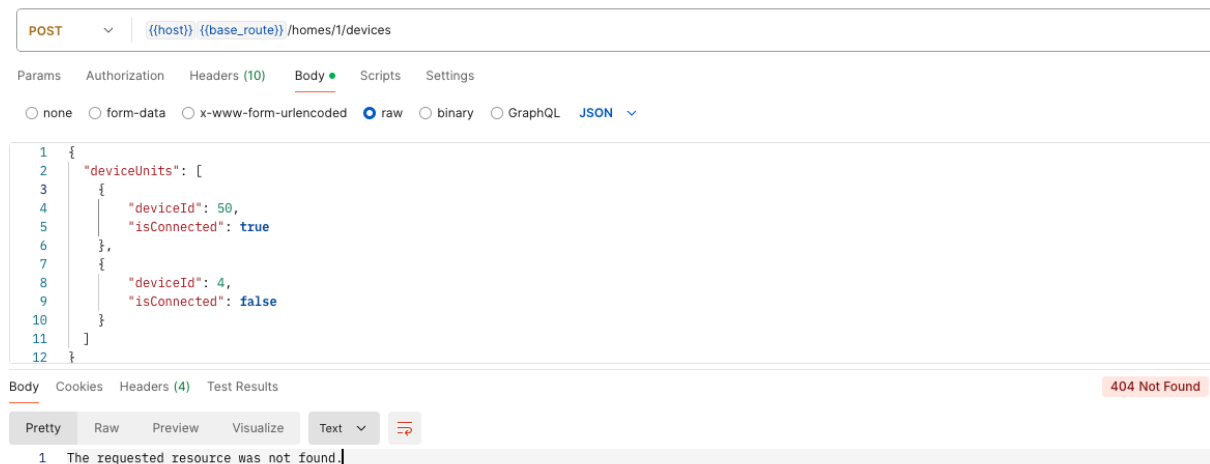
Response code:

404 Not Found

Response body:

“The requested resource was not found.”

Evidencia:



The screenshot shows a REST client interface. At the top, the method is POST and the URL is `{{host}} {{base_route}} /homes/1/devices`. Below the URL bar, there are tabs for Params, Authorization, Headers (10), Body, Scripts, and Settings. The Body tab is selected, and the content type is set to raw. The request body is a JSON object:

```
1 {
2   "deviceUnits": [
3     {
4       "deviceId": 50,
5       "isConnected": true
6     },
7     {
8       "deviceId": 4,
9       "isConnected": false
10    }
11  ]
12 }
```

At the bottom, there are tabs for Body, Cookies, Headers (4), and Test Results. The Body tab is selected, and the response is displayed in the Text view:

```
1 The requested resource was not found.
```

In the top right corner of the interface, a red status bar indicates "404 Not Found".

No se encuentra el hogar

Ruta

POST /api/v1/home/{id}/devices

Donde {id} es el Id de un hogar no existente en la plataforma.

Request headers OK

Request body OK

Response code:

404 Not Found

Response body:

“The requested resource was not found.”

Evidencia:

POST

⌵

{{host}} {{base_route}}/homes/10/devices

ParamsAuthorizationHeaders (10)Body●ScriptsSettings

☐ none

☐ form-data

☐ x-www-form-urlencoded

☒ raw

☐ binary

☐ GraphQL

JSON

⌵

1

{

2

"deviceUnits": [

3

{

4

"deviceId": 3,

5

"isConnected": true

6

},

7

{

8

"deviceId": 4,

9

"isConnected": false

10

}

11

]

12

}

BodyCookiesHeaders (4)Test Results

404 Not Found

Pretty

Raw

Preview

Visualize

Text

⌵

1 The requested resource was not found.