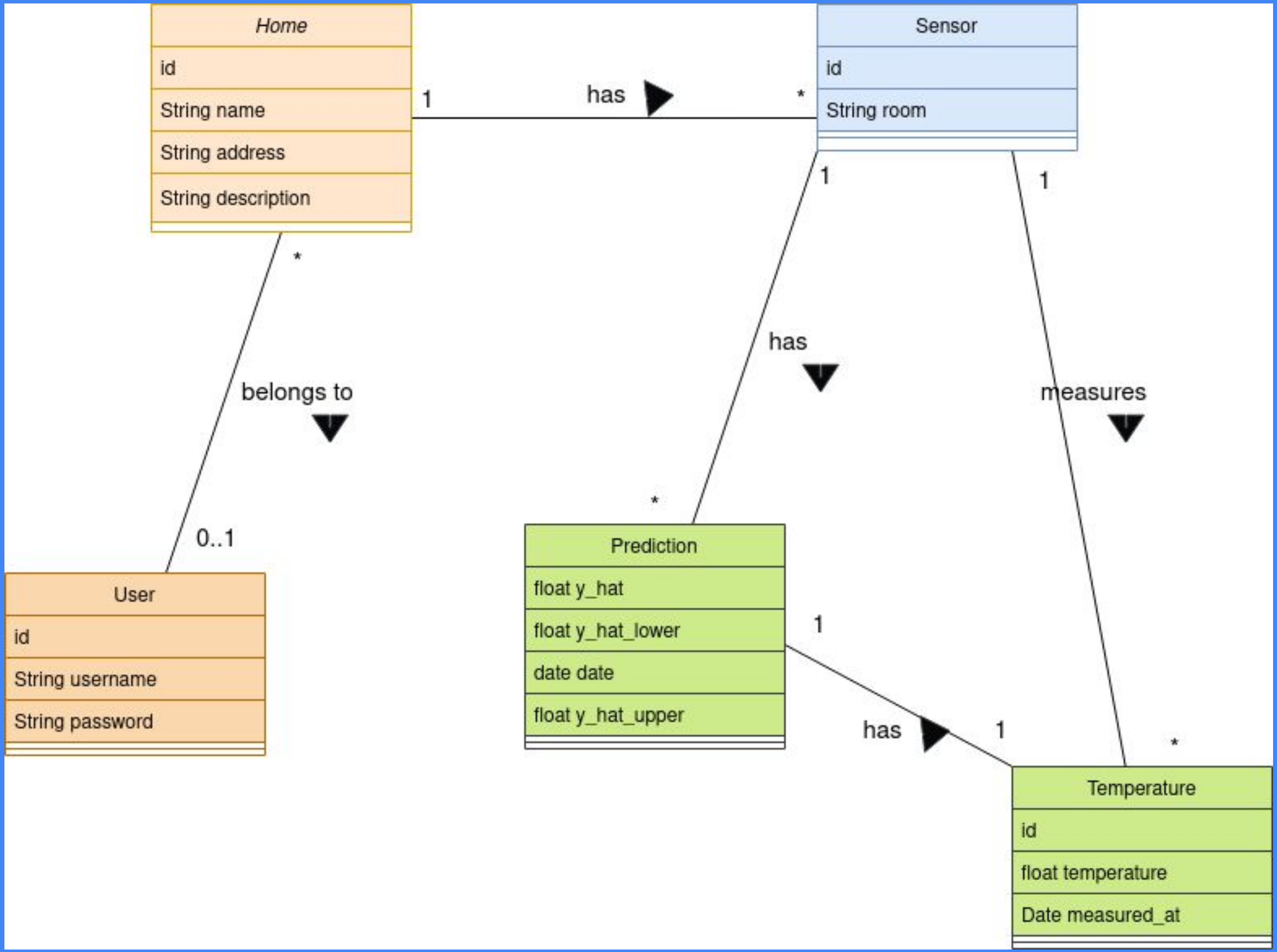
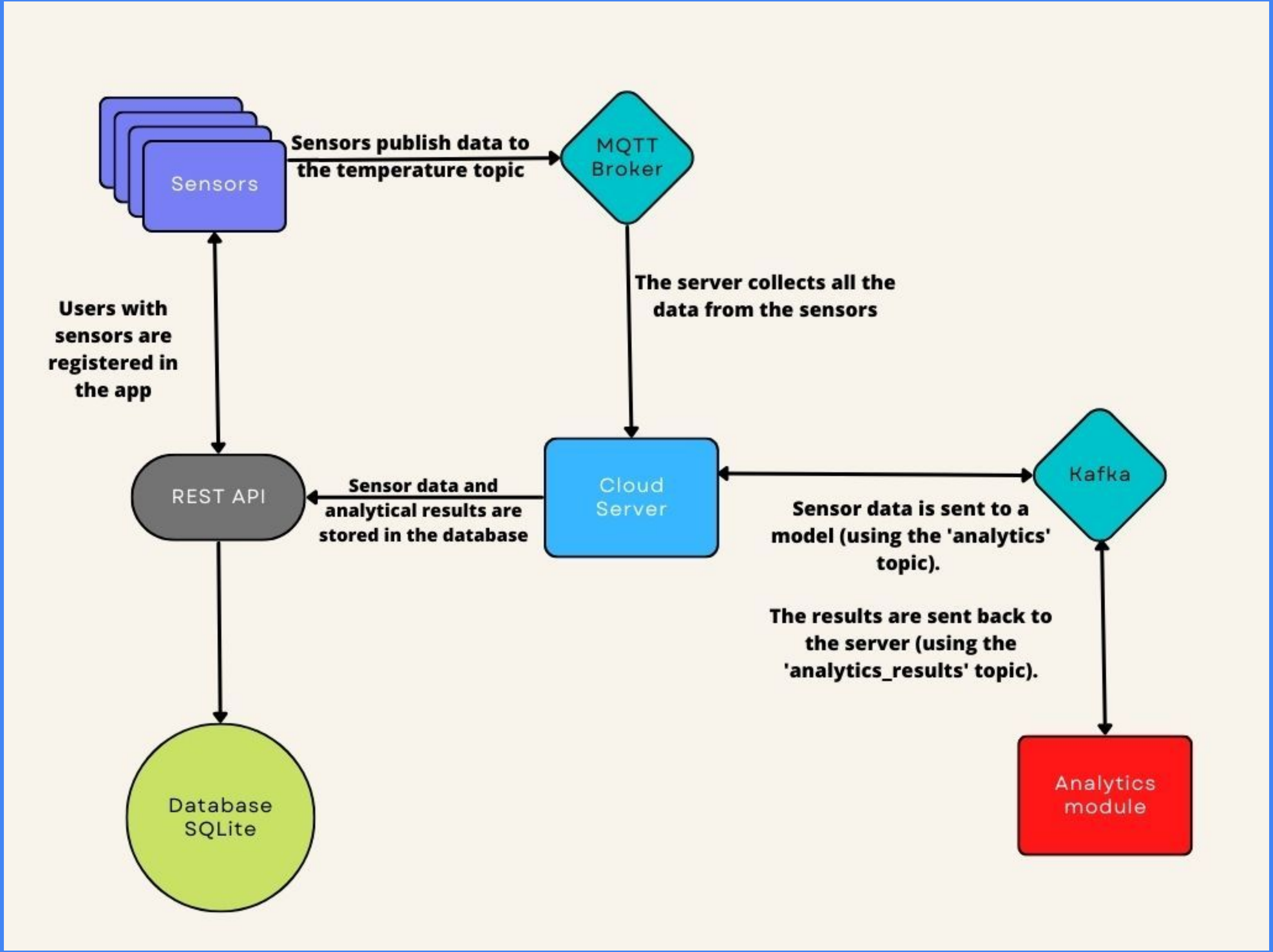
MyClimate API DEMO

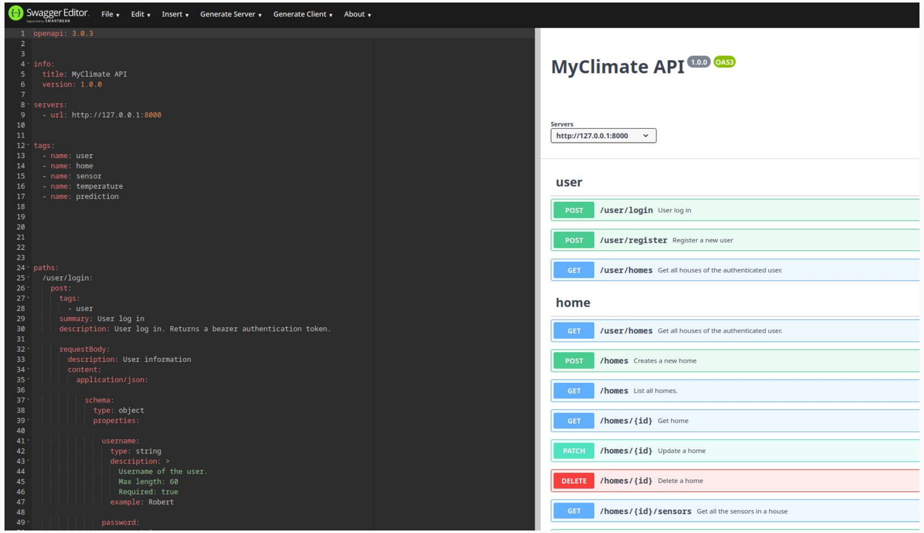
Marc Vivas Baiges


Home
id
String name
String address
String description
Sensor
id
String room
User
id
String username
String password
Prediction
float y_hat
float y_hat_lower
date date
float y_hat_upper
Temperature
id
float temperature
Date measured_at
has
belongs to
has
measures
has
1
*
*
0..1
1
*
1
*
1
1


Sensors
Sensors publish data to the temperature topic
MQTT Broker
The server collects all the data from the sensors
Users with sensors are registered in the app
REST API
Sensor data and analytical results are stored in the database
Cloud Server
Kafka
Sensor data is sent to a model (using the 'analytics' topic).
The results are sent back to the server (using the 'analytics_results' topic).
Database SQLite
Analytics module

API endpoints documentation

Open https://editor.swagger.io/ and import api_docs.yaml.



```yaml
openapi: 3.0.3


info:
  title: MyClimate API
  version: 1.0.0

servers:
  - url: http://127.0.0.1:8000


tags:
  - name: user
  - name: home
  - name: sensor
  - name: temperature
  - name: prediction






paths:
  /user/login:
    post:
      tags:
        - user
      summary: User log in
      description: User log in. Returns a bearer authentication token.

      requestBody:
        description: User information
        content:
          application/json:

            schema:
              type: object
              properties:

                username:
                  type: string
                  description: >
                    Username of the user.
                    Max length: 60
                    Required: true
                  example: Robert

                password:
```

MyClimate API 1.0.0 OAS3

Servers
http://127.0.0.1:8000

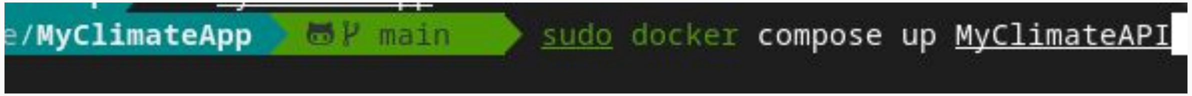
user

```
POST    /user/login      User log in
POST    /user/register   Register a new user
GET     /user/homes      Get all houses of the authenticated user.
```

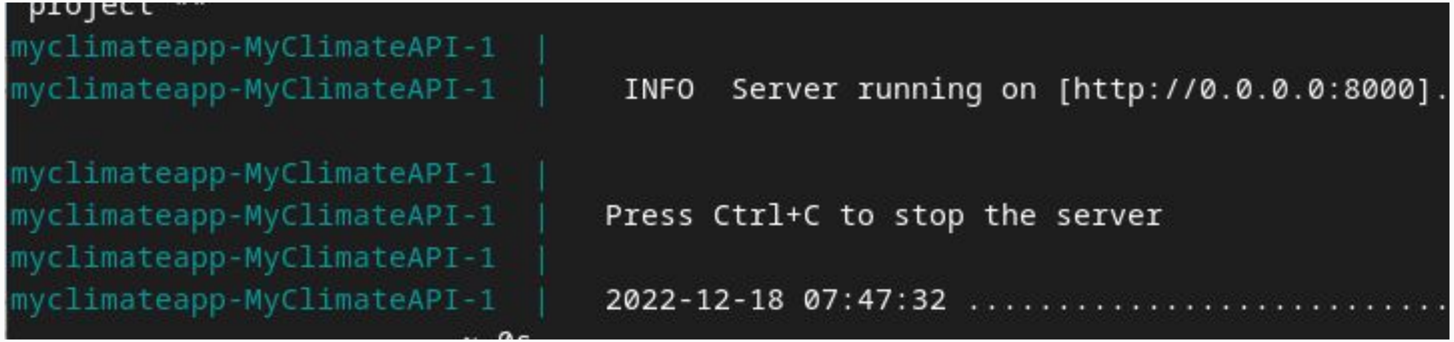
home

```
GET     /user/homes            Get all houses of the authenticated user.
POST    /homes                 Creates a new home
GET     /homes                 List all homes.
GET     /homes/{id}            Get home
PATCH   /homes/{id}            Update a home
DELETE  /homes/{id}            Delete a home
GET     /homes/{id}/sensors    Get all the sensors in a house
```

How to run the API?



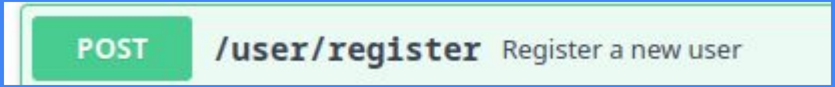
```
e/MyClimateApp   main   sudo docker compose up MyClimateAPI
```



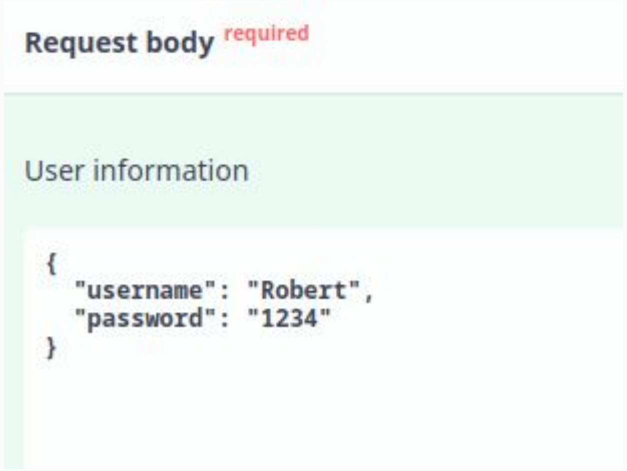
```
myclimateapp-MyClimateAPI-1  |
myclimateapp-MyClimateAPI-1  |     INFO  Server running on [http://0.0.0.0:8000].

myclimateapp-MyClimateAPI-1  |
myclimateapp-MyClimateAPI-1  |    Press Ctrl+C to stop the server
myclimateapp-MyClimateAPI-1  |
myclimateapp-MyClimateAPI-1  |    2022-12-18 07:47:32 ...........................
```

Web service demo

User register

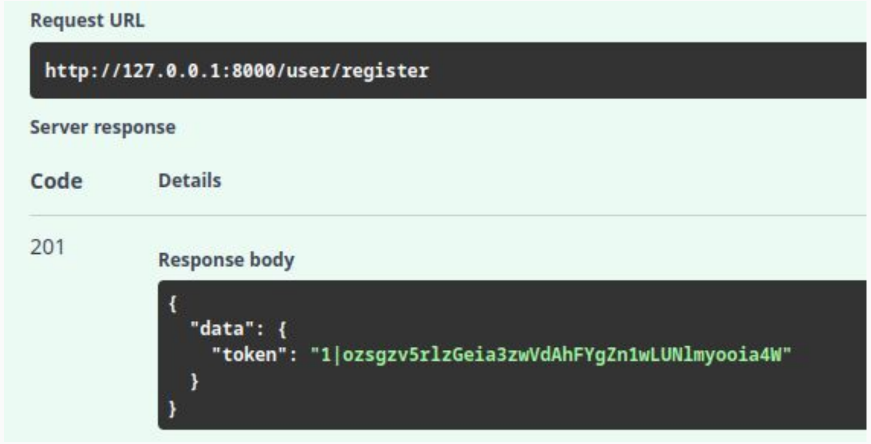

POST /user/register Register a new user

- All endpoints require an **authentication token** that can only be obtained when a new user is created or when a user logs in.


Request body required

User information

```
{
  "username": "Robert",
  "password": "1234"
}
```


Request URL

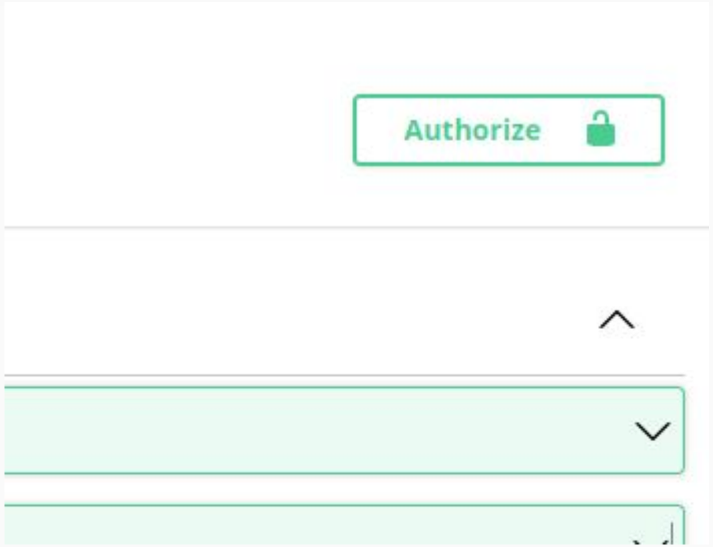
```
http://127.0.0.1:8000/user/register
```

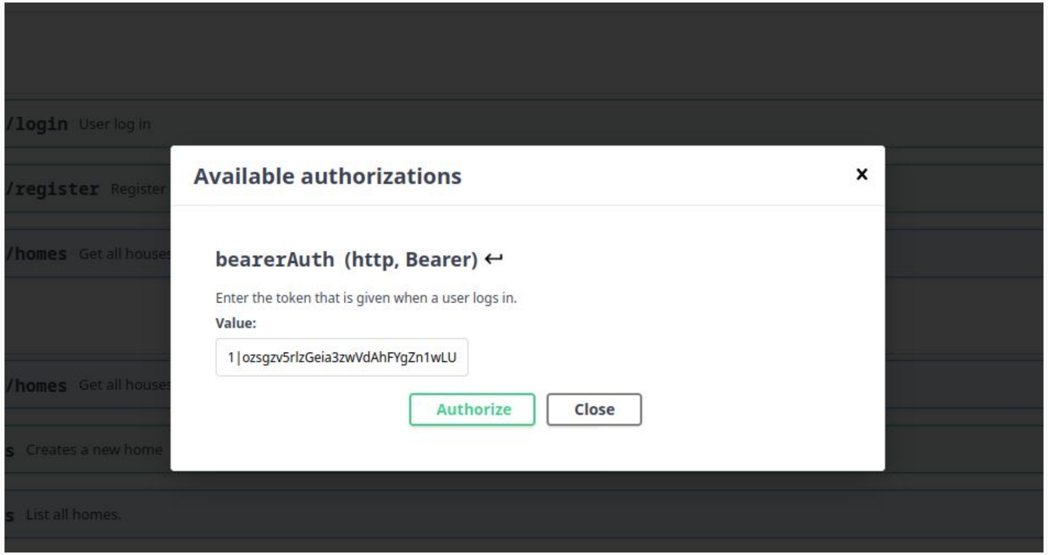
Server response

Code	Details
201	Response body

```
{
  "data": {
    "token": "1|ozsgzv5rlzGeia3zwVdAhFYgZn1wLUNlmyooia4W"
  }
}
```

Remember to authorize the requests!





Available authorizations

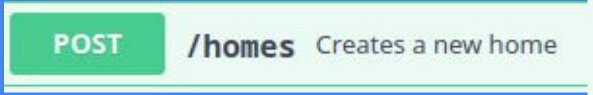
bearerAuth (http, Bearer)

Enter the token that is given when a user logs in.

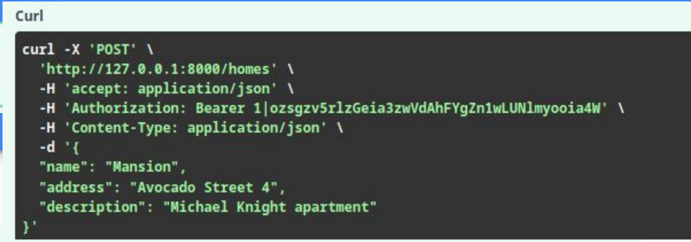
Value:

1|ozsgzv5rlzGeia3zwVdAhFYgZn1wLU

1. Create three different Homes with similar descriptions

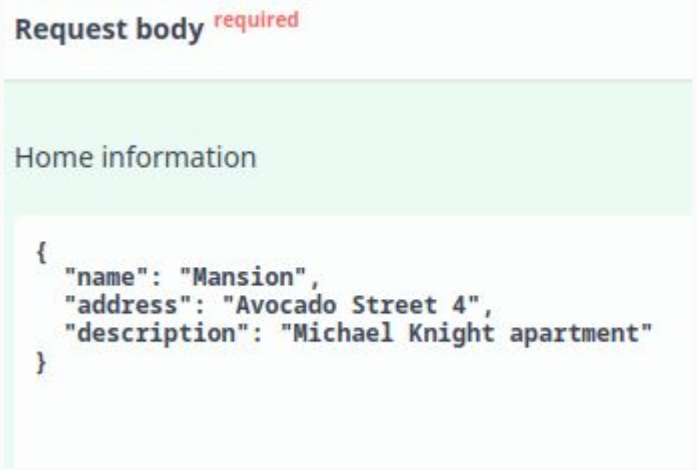


POST /homes Creates a new home



Curl

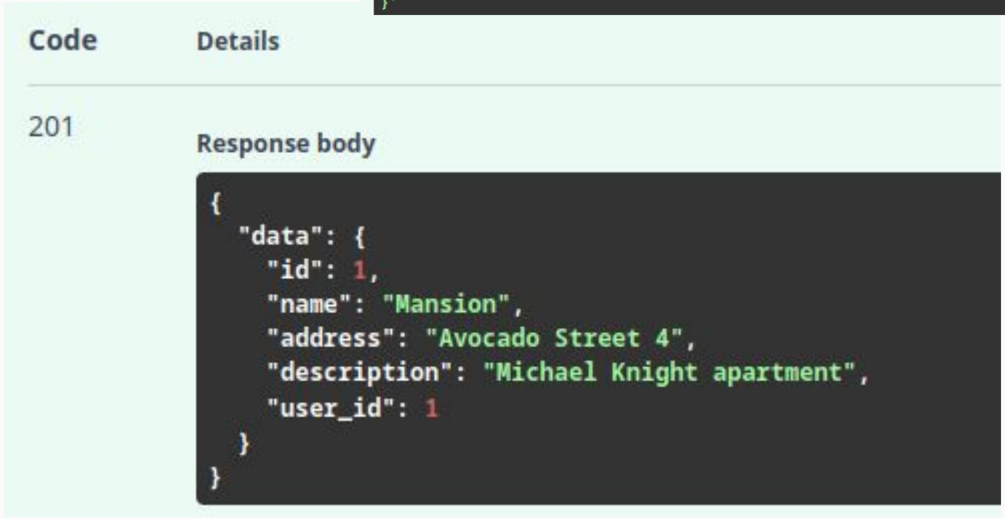
```
curl -X 'POST' \
  'http://127.0.0.1:8000/homes' \
  -H 'accept: application/json' \
  -H 'Authorization: Bearer 1|ozsgzv5rlzGeia3zwVdAhFYgZn1wLUNlmyooia4W' \
  -H 'Content-Type: application/json' \
  -d '{
  "name": "Mansion",
  "address": "Avocado Street 4",
  "description": "Michael Knight apartment"
}'
```



Request body required

Home information

```
{
  "name": "Mansion",
  "address": "Avocado Street 4",
  "description": "Michael Knight apartment"
}
```

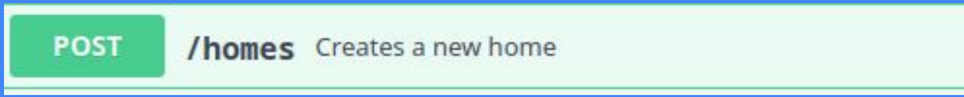


Code Details

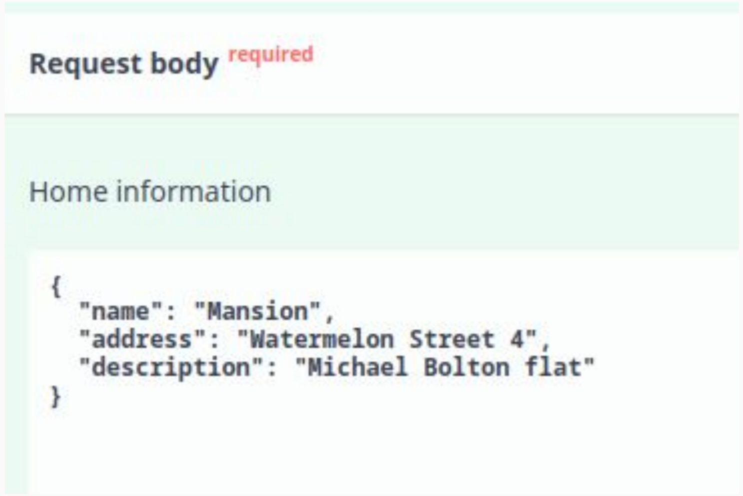
201

Response body

```
{
  "data": {
    "id": 1,
    "name": "Mansion",
    "address": "Avocado Street 4",
    "description": "Michael Knight apartment",
    "user_id": 1
  }
}
```

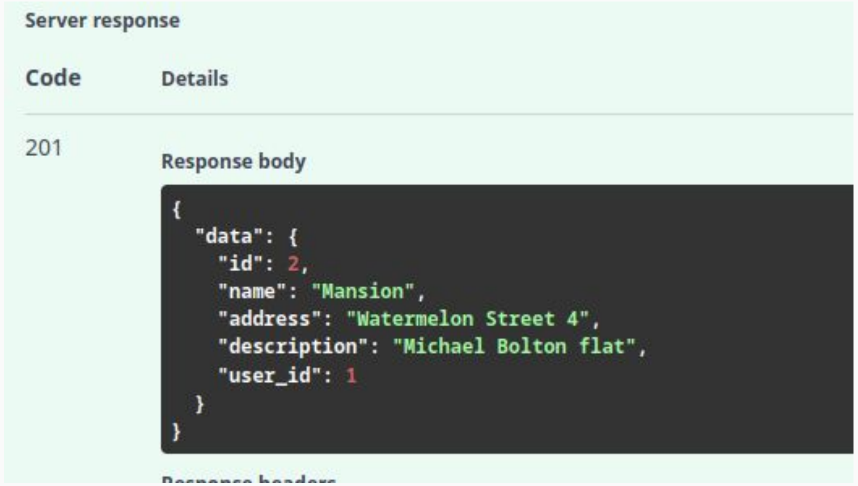

POST /homes Creates a new home

1. Create the second house


Request body required

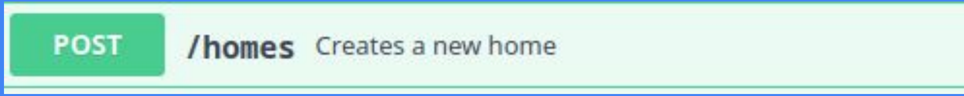
Home information

```
{
  "name": "Mansion",
  "address": "Watermelon Street 4",
  "description": "Michael Bolton flat"
}
```


Server response

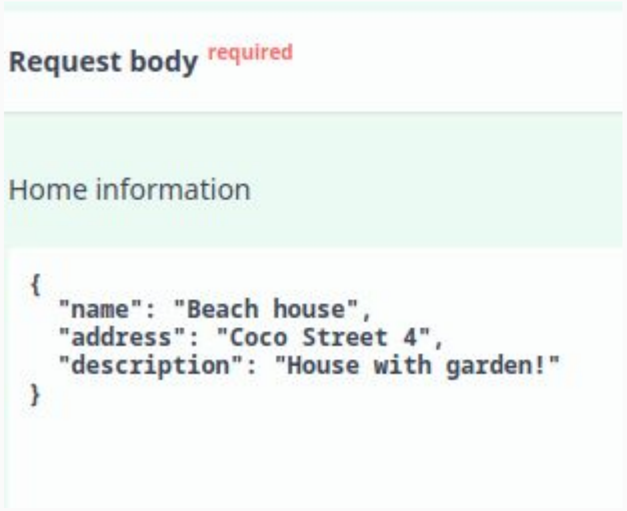
Code	Details
201	Response body

```
{
  "data": {
    "id": 2,
    "name": "Mansion",
    "address": "Watermelon Street 4",
    "description": "Michael Bolton flat",
    "user_id": 1
  }
}
```



POST /homes Creates a new home

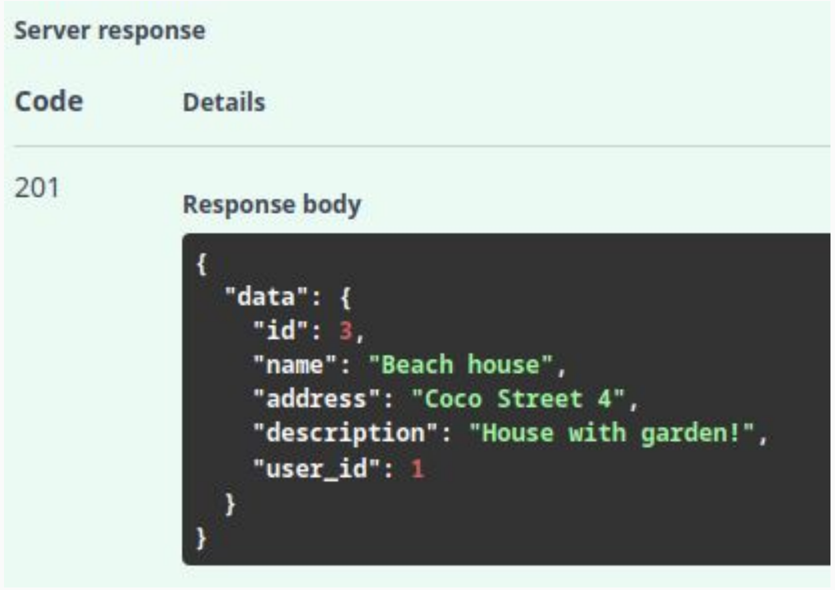
1. Create the third house



Request body required

Home information

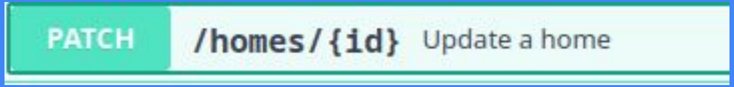
```
{
  "name": "Beach house",
  "address": "Coco Street 4",
  "description": "House with garden!"
}
```



Server response

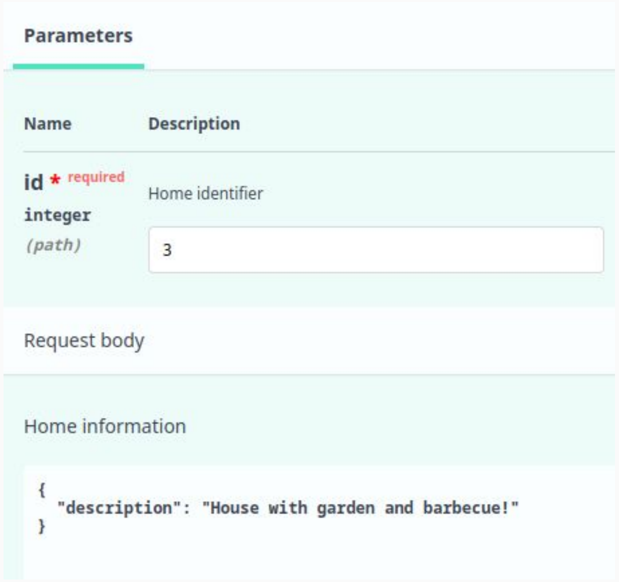
Code	Details
201	Response body

```
{
  "data": {
    "id": 3,
    "name": "Beach house",
    "address": "Coco Street 4",
    "description": "House with garden!",
    "user_id": 1
  }
}
```



PATCH /homes/{id} Update a home

2. Modify the description of the 3rd house



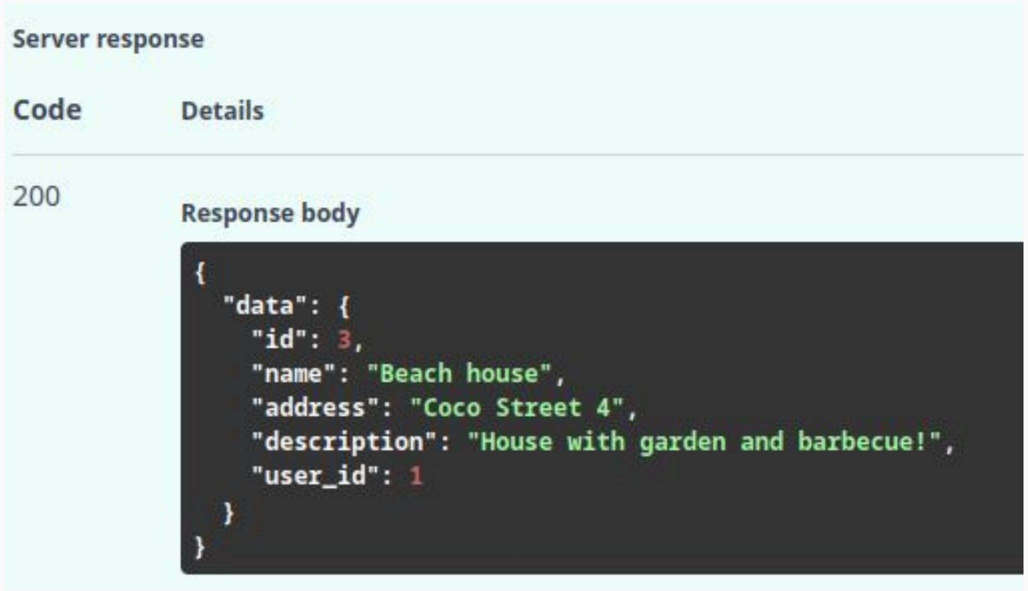
Parameters

Name	Description
id * required **integer** *(path)*	Home identifier 3

Request body

Home information

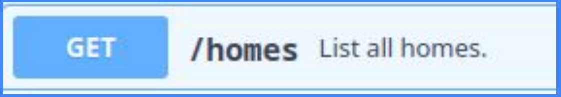
```
{
  "description": "House with garden and barbecue!"
}
```



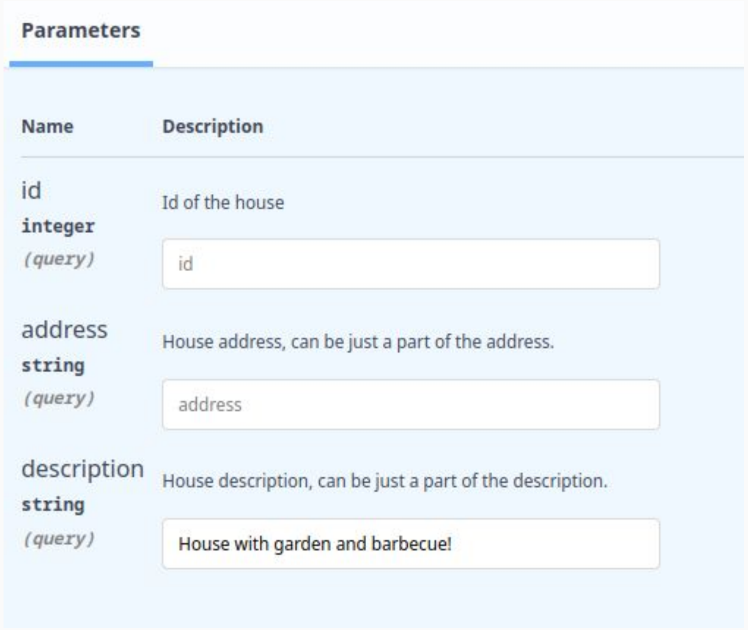
Server response

Code	Details
200	**Response body**

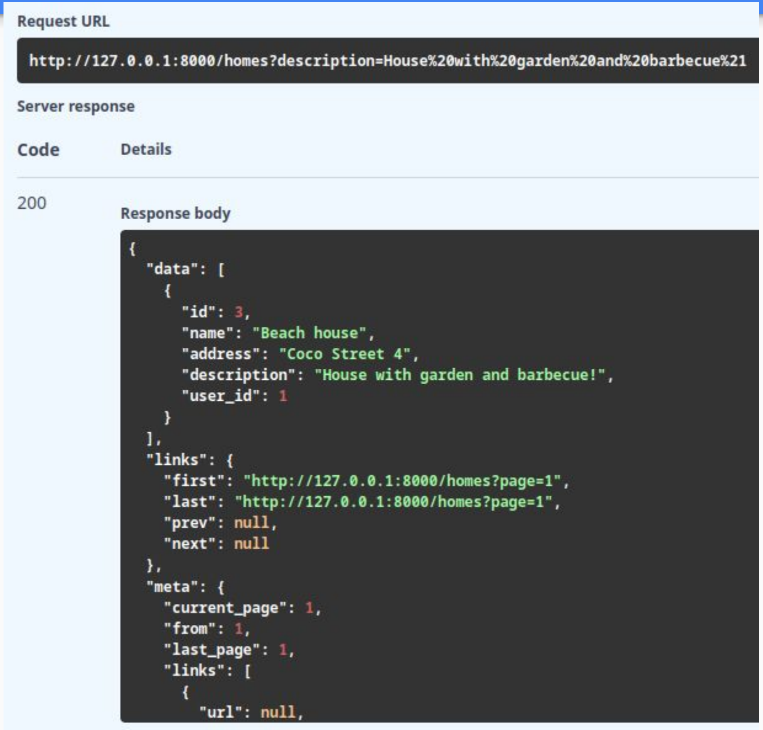
```
{
  "data": {
    "id": 3,
    "name": "Beach house",
    "address": "Coco Street 4",
    "description": "House with garden and barbecue!",
    "user_id": 1
  }
}
```


GET /homes List all homes.

3. Search a house by its description (full)


Parameters

Name	Description
id integer *(query)*	Id of the house id
address string *(query)*	House address, can be just a part of the address. address
description string *(query)*	House description, can be just a part of the description. House with garden and barbecue!


Request URL

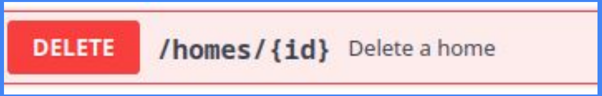
```
http://127.0.0.1:8000/homes?description=House%20with%20garden%20and%20barbecue%21
```

Server response

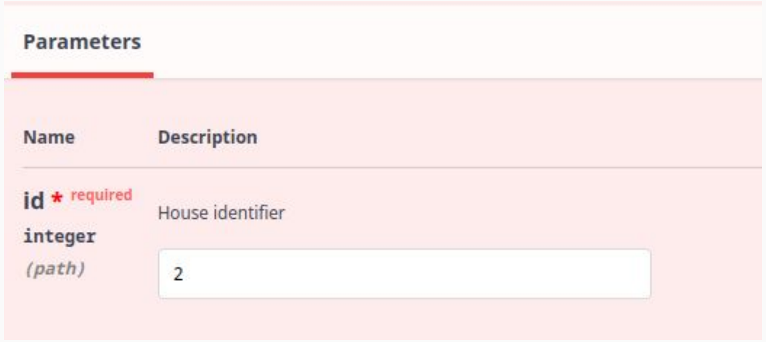
Code Details

200 Response body

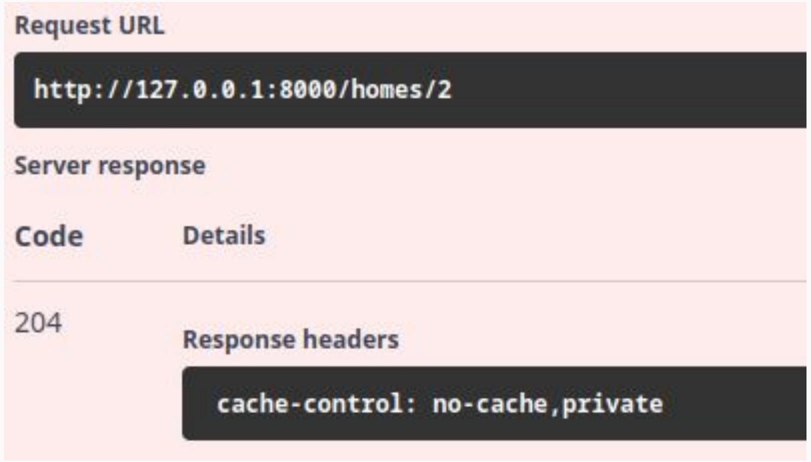
```
{
  "data": [
    {
      "id": 3,
      "name": "Beach house",
      "address": "Coco Street 4",
      "description": "House with garden and barbecue!",
      "user_id": 1
    }
  ],
  "links": {
    "first": "http://127.0.0.1:8000/homes?page=1",
    "last": "http://127.0.0.1:8000/homes?page=1",
    "prev": null,
    "next": null
  },
  "meta": {
    "current_page": 1,
    "from": 1,
    "last_page": 1,
    "links": [
      {
        "url": null,
```


DELETE /homes/{id} Delete a home

4. Delete the second house created


Parameters

Name	Description
id * required integer *(path)*	House identifier 2

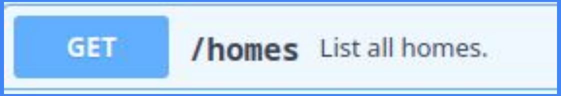

Request URL

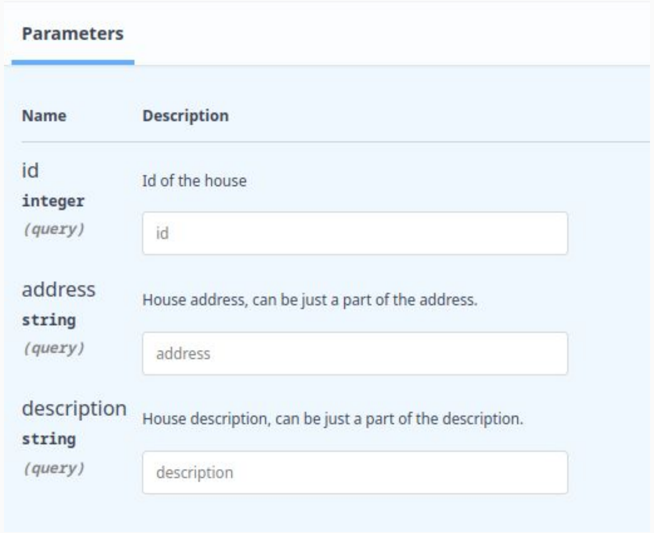
```
http://127.0.0.1:8000/homes/2
```

Server response

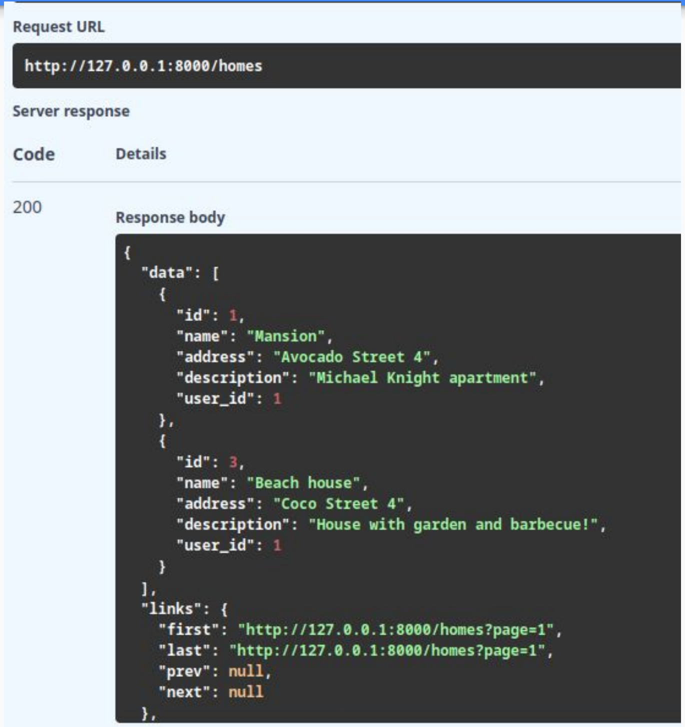
Code	Details
204	**Response headers** `cache-control: no-cache,private`

5. List all homes


GET /homes List all homes.


Parameters

Name	Description
id **integer** *(query)*	Id of the house id
address **string** *(query)*	House address, can be just a part of the address. address
description **string** *(query)*	House description, can be just a part of the description. description


Request URL

```
http://127.0.0.1:8000/homes
```

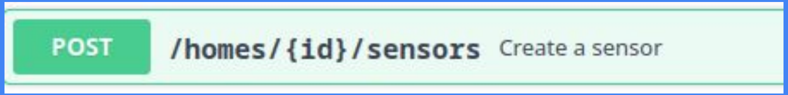
Server response

Code | Details

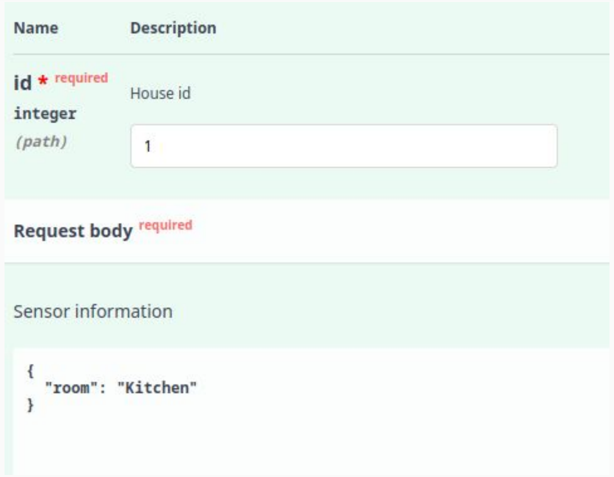
200

Response body

```
{
  "data": [
    {
      "id": 1,
      "name": "Mansion",
      "address": "Avocado Street 4",
      "description": "Michael Knight apartment",
      "user_id": 1
    },
    {
      "id": 3,
      "name": "Beach house",
      "address": "Coco Street 4",
      "description": "House with garden and barbecue!",
      "user_id": 1
    }
  ],
  "links": {
    "first": "http://127.0.0.1:8000/homes?page=1",
    "last": "http://127.0.0.1:8000/homes?page=1",
    "prev": null,
    "next": null
  },
```


POST /homes/{id}/sensors Create a sensor

6. Create two sensors in the first House

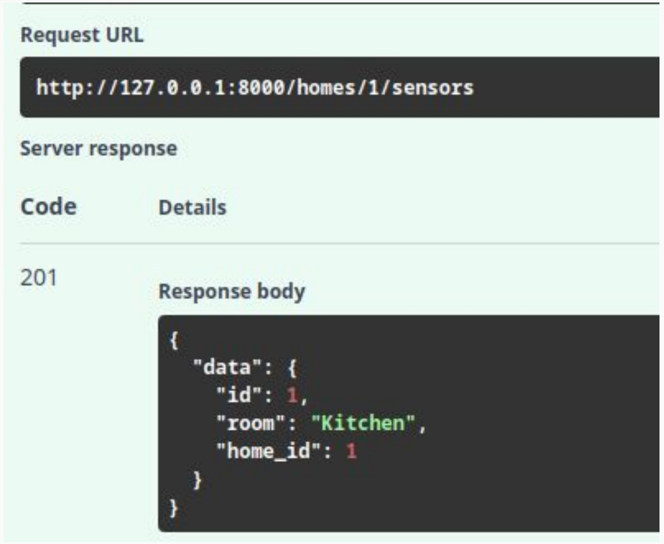


Name	Description
id * required integer *(path)*	House id 1

Request body required

Sensor information

```
{
  "room": "Kitchen"
}
```



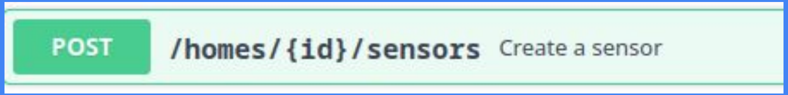
Request URL

```
http://127.0.0.1:8000/homes/1/sensors
```

Server response

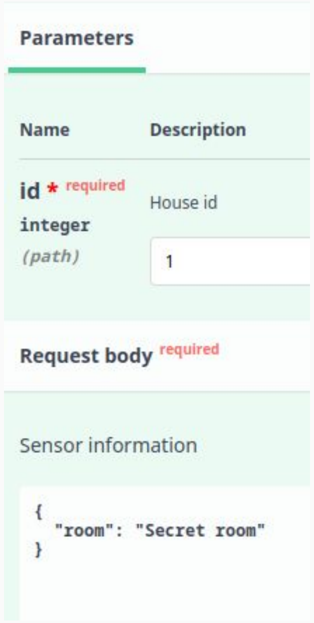
Code	Details
201	Response body

```
{
  "data": {
    "id": 1,
    "room": "Kitchen",
    "home_id": 1
  }
}
```



POST /homes/{id}/sensors Create a sensor

6. Create two sensors in the first House



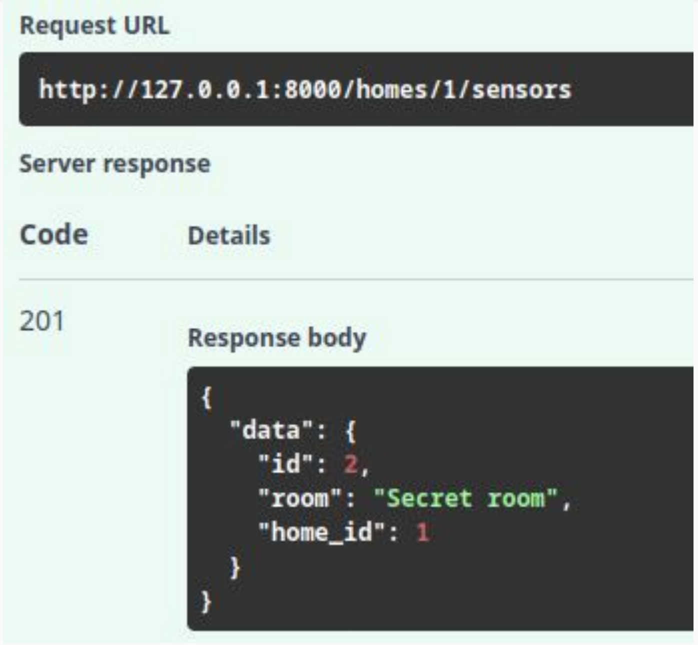
Parameters

Name	Description
id * required integer *(path)*	House id 1

Request body required

Sensor information

```
{
  "room": "Secret room"
}
```



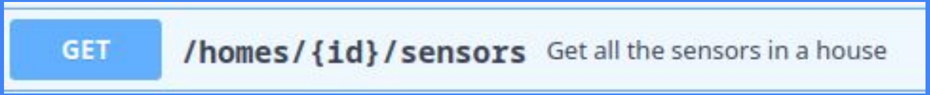
Request URL

```
http://127.0.0.1:8000/homes/1/sensors
```

Server response

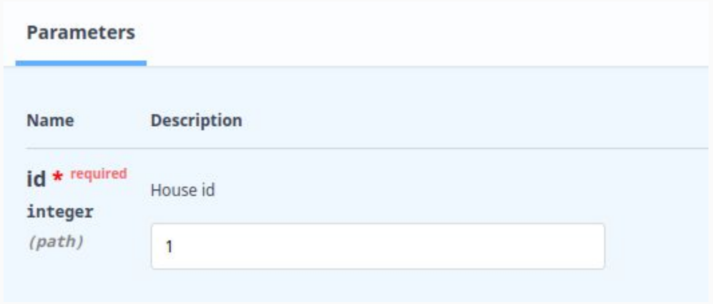
Code	Details
201	Response body

```
{
  "data": {
    "id": 2,
    "room": "Secret room",
    "home_id": 1
  }
}
```



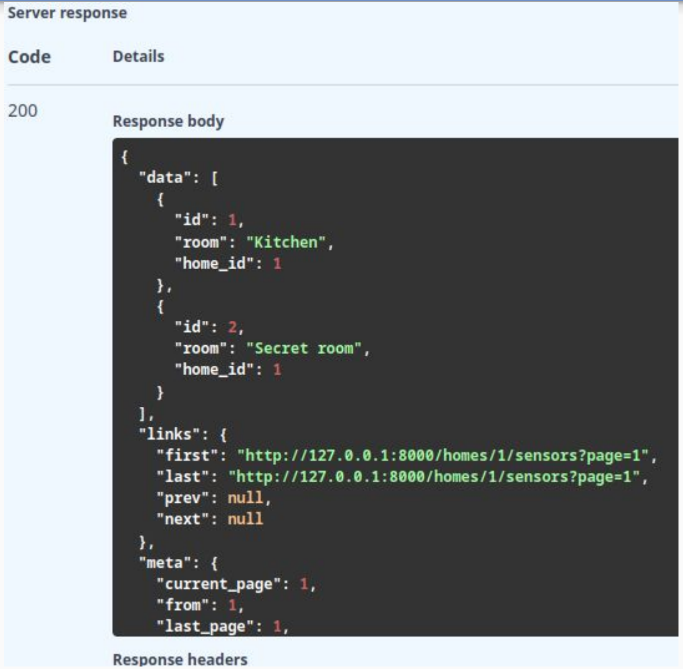
GET /homes/{id}/sensors Get all the sensors in a house

7. List first house sensors



Parameters

Name	Description
id * required integer *(path)*	House id 1

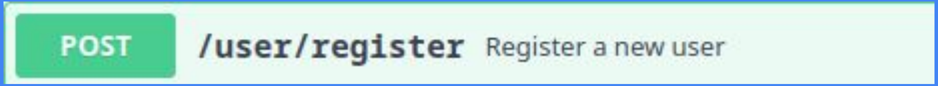


Server response

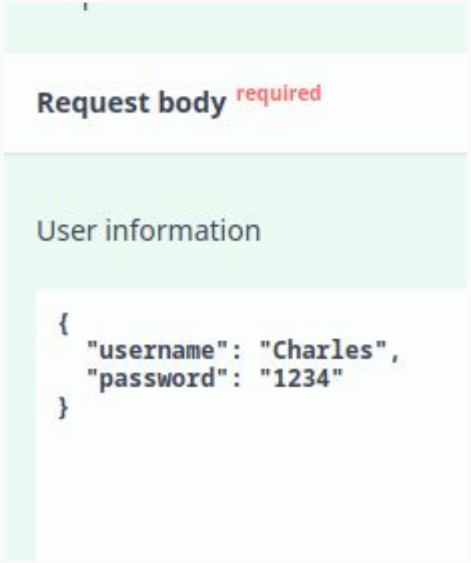
Code	Details
200	Response body

```
{
  "data": [
    {
      "id": 1,
      "room": "Kitchen",
      "home_id": 1
    },
    {
      "id": 2,
      "room": "Secret room",
      "home_id": 1
    }
  ],
  "links": {
    "first": "http://127.0.0.1:8000/homes/1/sensors?page=1",
    "last": "http://127.0.0.1:8000/homes/1/sensors?page=1",
    "prev": null,
    "next": null
  },
  "meta": {
    "current_page": 1,
    "from": 1,
    "last_page": 1,
```

Response headers

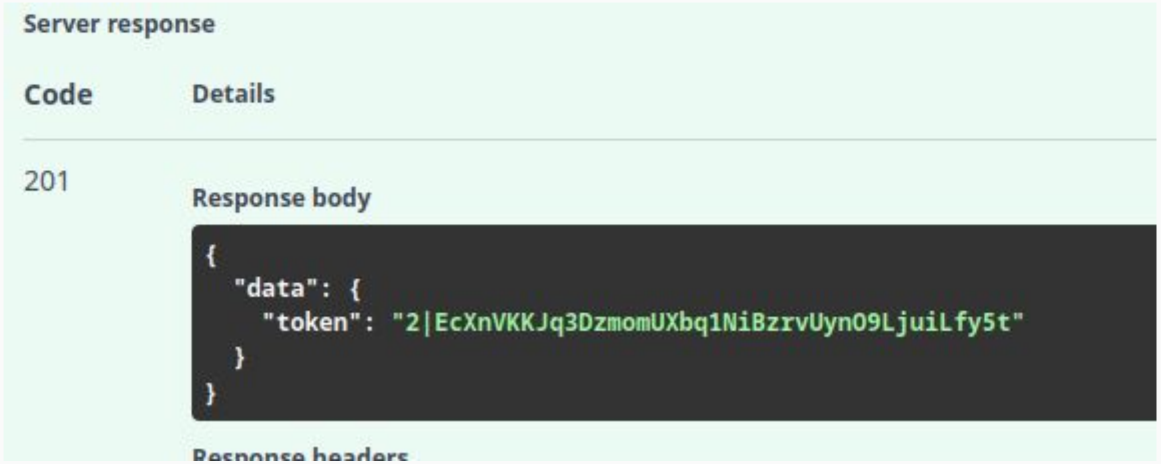

POST /user/register Register a new user

8. Create another user


Request body required

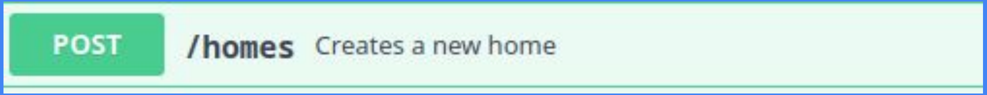
User information

```
{
  "username": "Charles",
  "password": "1234"
}
```


Server response

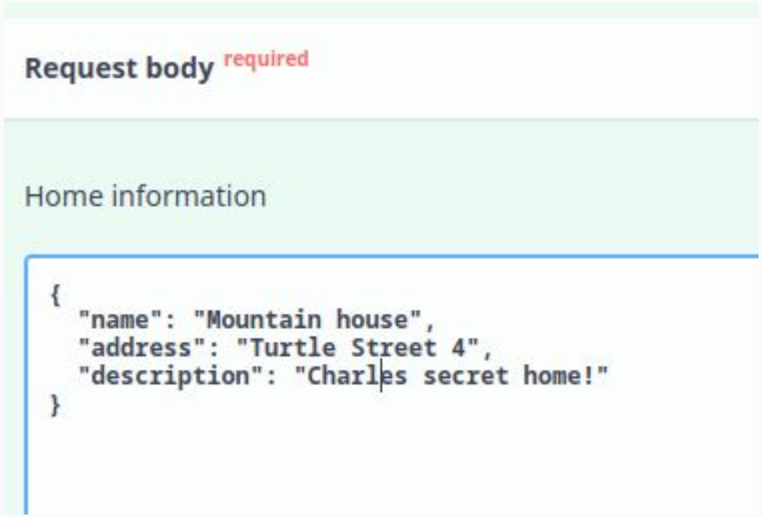
Code	Details
201	Response body

```
{
  "data": {
    "token": "2|EcXnVKKJq3DzmomUXbq1NiBzrvUyn09LjuiLfy5t"
  }
}
```



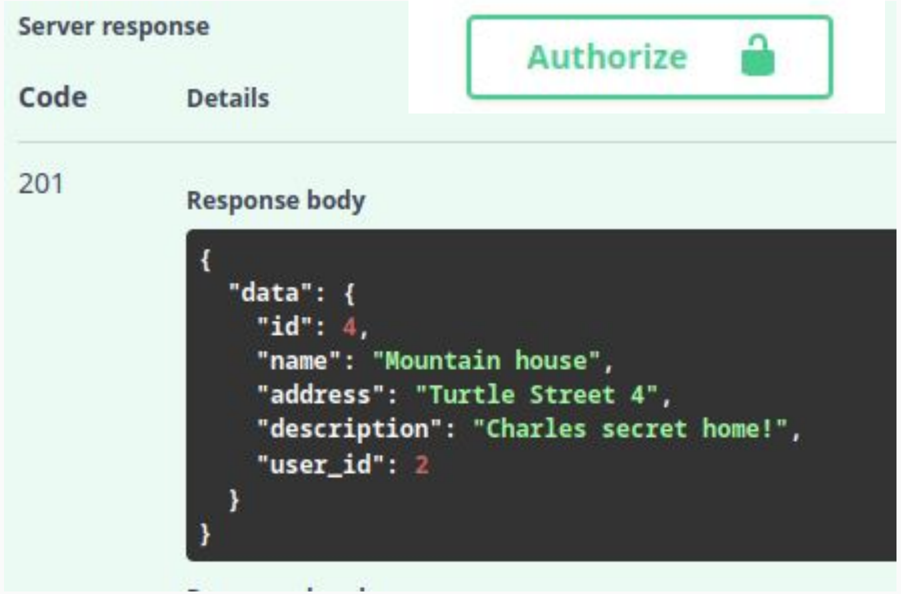
/homes Creates a new home

9. Upload a home using the new user


Request body

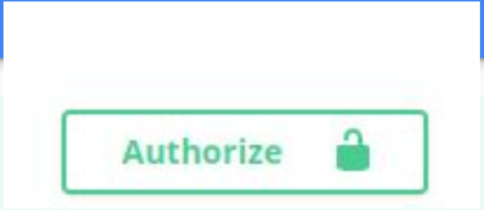
Home information

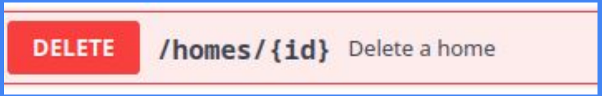
```
{
  "name": "Mountain house",
  "address": "Turtle Street 4",
  "description": "Charles secret home!"
}
```


Server response

Code	Details
201	Response body

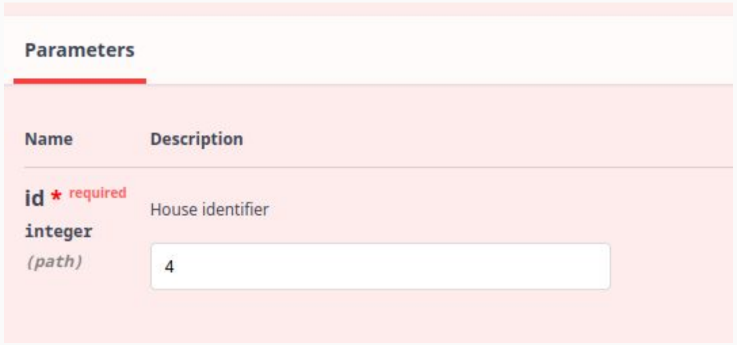
```
{
  "data": {
    "id": 4,
    "name": "Mountain house",
    "address": "Turtle Street 4",
    "description": "Charles secret home!",
    "user_id": 2
  }
}
```



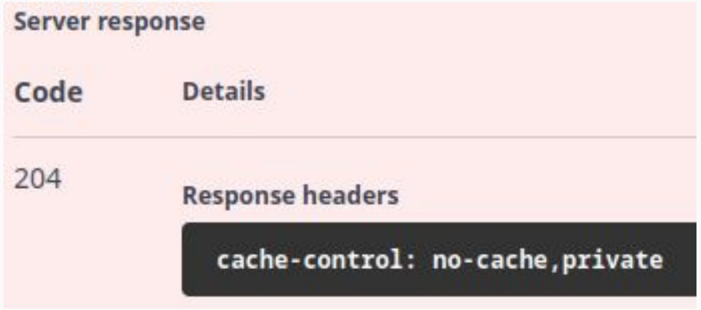

DELETE /homes/{id} Delete a home

10. Delete the uploaded home of the new user

Only the owner of the house is allowed to delete it.



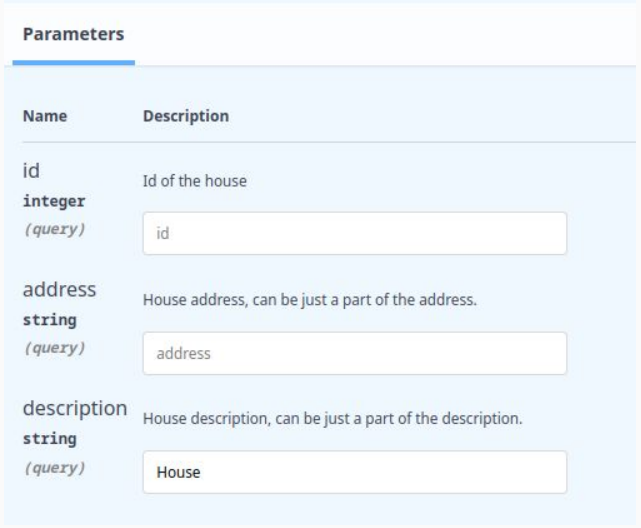
Name	Description
id * required integer *(path)*	House identifier 4


Server response

Code	Details
204	Response headers

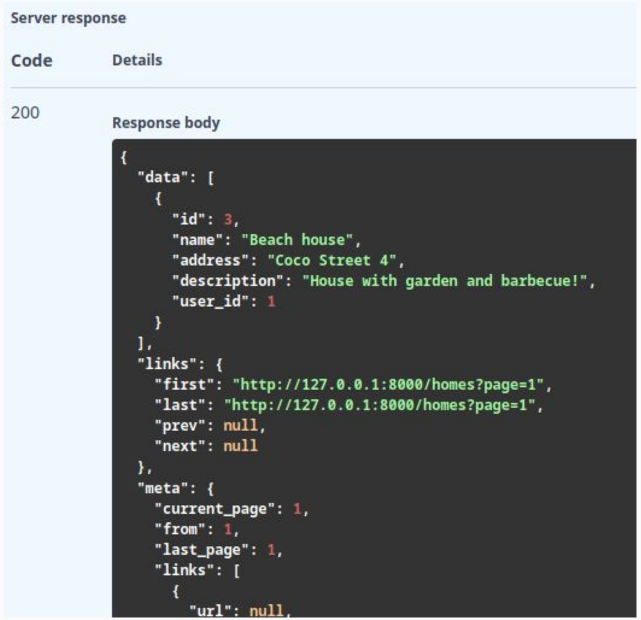
```
cache-control: no-cache,private
```

11. Search Home by its description (partial)



Parameters

Name	Description
id **integer** *(query)*	Id of the house id
address **string** *(query)*	House address, can be just a part of the address. address
description **string** *(query)*	House description, can be just a part of the description. House



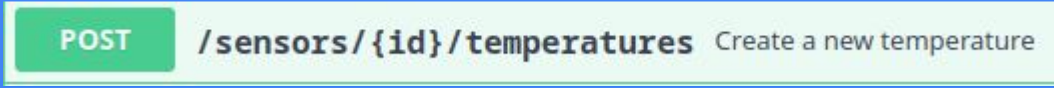
Server response

Code	Details
200	Response body

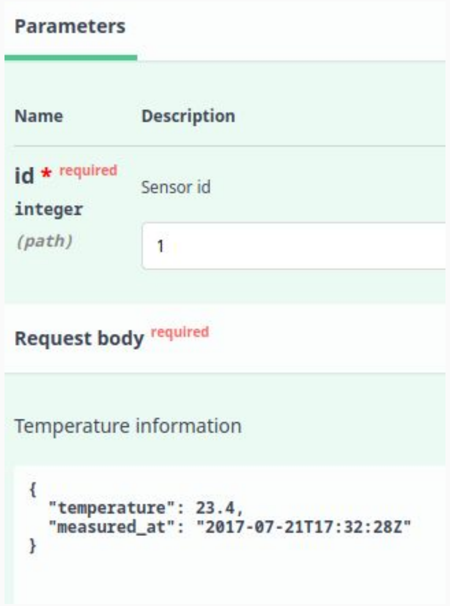
```
{
  "data": [
    {
      "id": 3,
      "name": "Beach house",
      "address": "Coco Street 4",
      "description": "House with garden and barbecue!",
      "user_id": 1
    }
  ],
  "links": {
    "first": "http://127.0.0.1:8000/homes?page=1",
    "last": "http://127.0.0.1:8000/homes?page=1",
    "prev": null,
    "next": null
  },
  "meta": {
    "current_page": 1,
    "from": 1,
    "last_page": 1,
    "links": [
      {
        "url": null,
```

12. Show its user (Not implemented)

13. Create a new temperature measured by the first sensor of user 1.



POST /sensors/{id}/temperatures Create a new temperature



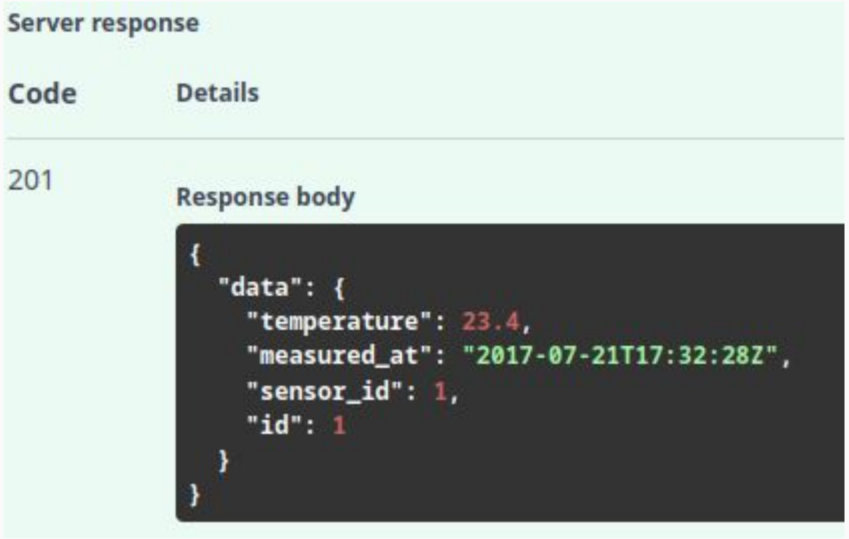
Parameters

Name	Description
id * required integer *(path)*	Sensor id 1

Request body required

Temperature information

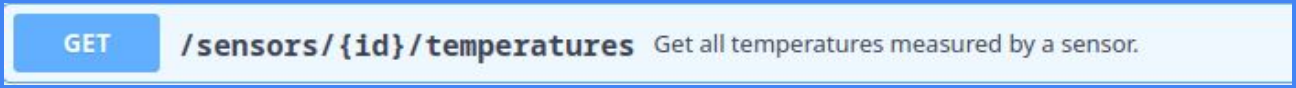
```
{
  "temperature": 23.4,
  "measured_at": "2017-07-21T17:32:28Z"
}
```



Server response

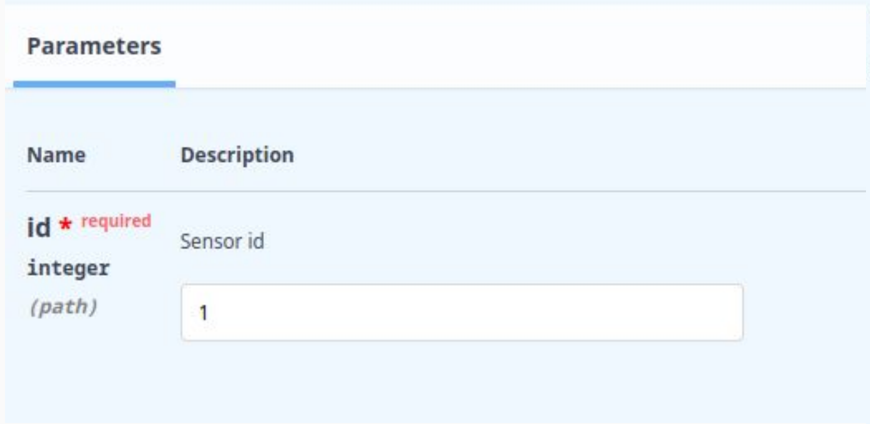
Code	Details
201	Response body

```
{
  "data": {
    "temperature": 23.4,
    "measured_at": "2017-07-21T17:32:28Z",
    "sensor_id": 1,
    "id": 1
  }
}
```



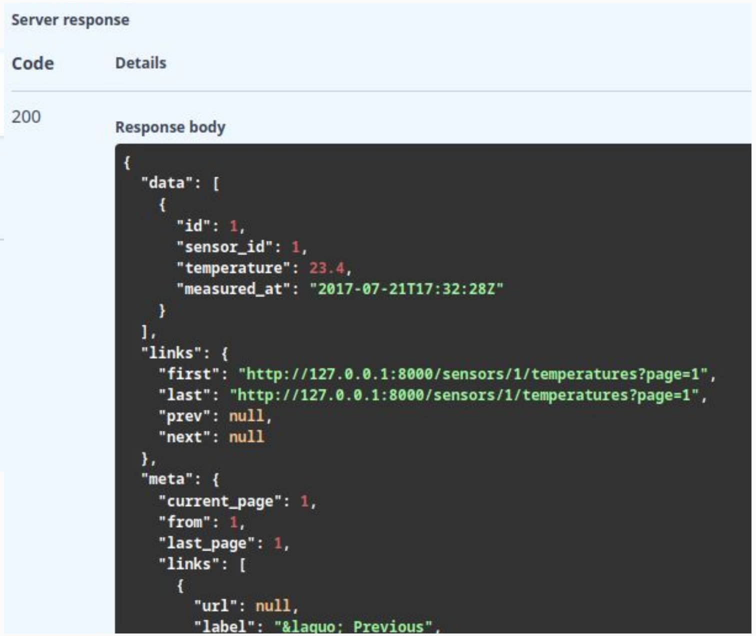
GET /sensors/{id}/temperatures Get all temperatures measured by a sensor.

13. Show the temperatures of sensor 1



Parameters

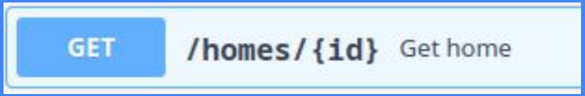
Name	Description
id * required integer *(path)*	Sensor id 1



Server response

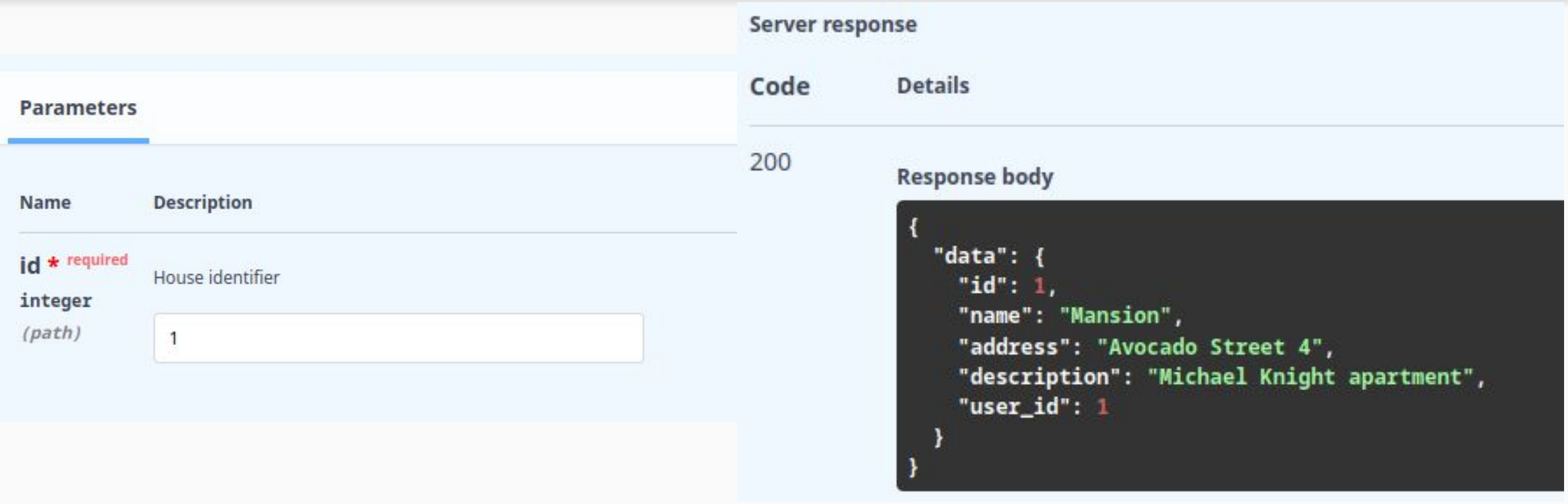
Code	Details
200	Response body

```
{
  "data": [
    {
      "id": 1,
      "sensor_id": 1,
      "temperature": 23.4,
      "measured_at": "2017-07-21T17:32:28Z"
    }
  ],
  "links": {
    "first": "http://127.0.0.1:8000/sensors/1/temperatures?page=1",
    "last": "http://127.0.0.1:8000/sensors/1/temperatures?page=1",
    "prev": null,
    "next": null
  },
  "meta": {
    "current_page": 1,
    "from": 1,
    "last_page": 1,
    "links": [
      {
        "url": null,
        "label": "&laquo; Previous",
```



GET `/homes/{id}` Get home

14. Check some ID from an existing home



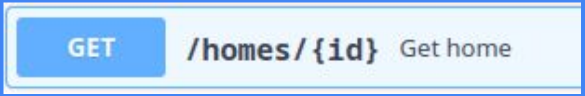
Parameters

Name	Description
id * required integer *(path)*	House identifier 1

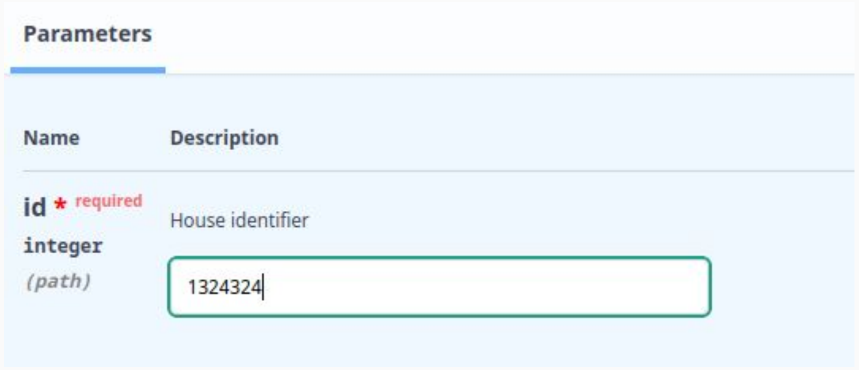
Server response

Code	Details
200	Response body

```
{
  "data": {
    "id": 1,
    "name": "Mansion",
    "address": "Avocado Street 4",
    "description": "Michael Knight apartment",
    "user_id": 1
  }
}
```

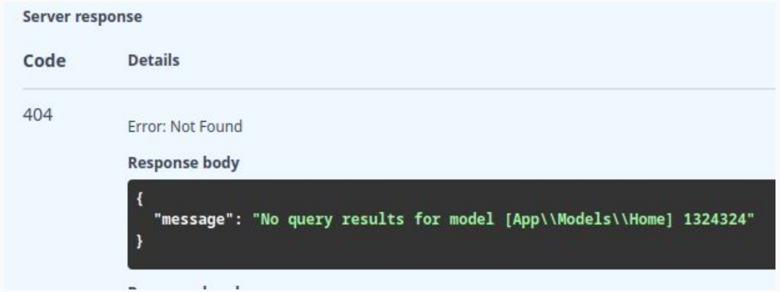


15. Check some other nonexistent ID



id * required
integer
(path)
House identifier

1324324


Server response

Code	Details
404	Error: Not Found

Response body

```
{
  "message": "No query results for model [App\\Models\\Home] 1324324"
}
```

Marc Vivas Baiges

More documentation in README.md and API_README.md