**Backend Developer Test**

**Analytic Lab Works**

This is a technical test for applying to backend position. The solution must be developed in Java and published to a GitHub repository.

1. What data structures and algorithms can be used for storing and filtering by pattern (or regex) a set of strings? Write the code using Java 8 functions and Stream API.

*R/*

*The data structure that can be used is a collection, this from different functions allows mapping the data, which means passing element by elements to perform a specific transformation or function. Filter the elements from a specific condition that in the case of strings can be regular expressions and finally an order.*

*The code is in the file Pregunta1.java*

1. Write a function that returns the count of distinct case-insensitive alphabetic characters and numeric digits occurring more than once in an input string. Example:
   1. "abcde" => {}
   2. "aabbcde" => { ‘a’: 2, ‘b’: 2 }
   3. "aabBcde" => { ‘a’: 2, ‘b’: 2 }
   4. "indivisibility" => { ‘i’: 6 }
   5. "Indivisibilities" => { ‘i’: 6, ‘s’: 2 }
   6. "aA11" => { ‘a’: 2, 1: 2 }
   7. "ABBA" => { ‘a’: 2, ‘b’: 2 }

*R/ the code is in the file Pregunta2.java*

1. Given the entity relationship diagram below, code SQL sentences for:
   1. Get all buses for “Concessionaire 1”.

R/

SELECT buses.\*

FROM bus as buses

INNER JOIN concessionaire as conce

ON buses.concessionaireId = conce.Id

WHERE conce.name = 'Concessionaire 1'

* 1. Get all NVR devices for buses with type equal to “Bi-articulado”.

SELECT devi.\*

FROM bus as buses

INNER JOIN device as devi

ON devi.busId = buses.Id

INNER JOIN devicetype as deviType

ON devi.deviceTypeId = deviType.id

WHERE deviType.name = 'NVR' AND buses.type = 'Bi-articulado'

* 1. Summarize the quantity of devices by status (Active / Inactive) and bus motor (Diesel / Gas / Electric / Hybrid).

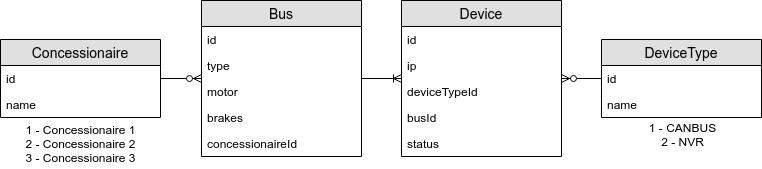
SELECT devi.id , devi.status , buses.motor , count(devi.id)

FROM bus as buses

INNER JOIN device as devi

ON devi.busId = buses.Id

GROUP BY devi.id , devi.status , buses.motor



The files whit Script for Querys and create DB are In the folder Buses\src\main\java\com\buses\Buses\Sql

1. Design and code an API REST for accessing the resources in the above database.
   1. What HTTP endpoints and methods would you enable for creation, reading, modification and deletion?

Base URL: <http://127.0.0.1:8080/api>

Concessionaire:

GET /concessionaire

Response:

[

    {

        "id": Integer,

        "name": String

    }

]

GET /concessionaire/{id}

Params:

Id: concesseionaire ID

Response:

    {

        "id": Integer,

        "name": String

    }

POST / concessionaire

Body(json):

{"name": String}

Response:

    {

        "id": Integer,

        "name": String

    }

PUT / concessionaire

Body(json):

{ id": Integer,

        "name": String }

Response:

    {

        "id": Integer,

        "name": String

    }

DELETE /concessionaire/{id}

Params:

Id: concesseionaire ID

Response:

    {

        "message": String

    }

DeviceType:

GET /deviceType

Response:

[

    {

        "id": Integer,

        "name": String

    }

]

GET / deviceType /{id}

Params:

Id: deviceType ID

Response:

    {

        "id": Integer,

        "name": String

    }

POST / deviceType

Body(json):

{"name": String}

Response:

    {

        "id": Integer,

        "name": String

    }

PUT / deviceType

Body(json):

{ id": Integer,

        "name": String }

Response:

    {

        "id": Integer,

        "name": String

    }

DELETE / deviceType /{id}

Params:

Id: deviceType ID

Response:

    {

        "message": String

    }

Buses:

GET /buses

Response:

[

  {

        "id": Intener,

        "type": String

        "motor": String

        "brakes": String

        "concessionaireId": Concessionaire

    }

]

GET / buses /{id}

Params:

Id: bus ID

Response:

    {

        "id": Intener,

        "type": String

        "motor": String

        "brakes": String

        "concessionaireId": Concessionaire

    }

POST / buses

Body(json):

    {

        "type": String

        "motor": String

        "brakes": String

        "concessionaireId": Concessionaire

    }

Response:

    {

        "id": Intener,

        "type": String

        "motor": String

        "brakes": String

        "concessionaireId": Concessionaire

    }

PUT / buses

Body(json):

{  "id": Intener,

        "type": String

        "motor": String

        "brakes": String

        "concessionaireId": Concessionaire

}

Response:

    {

        "id": Intener,

        "type": String

        "motor": String

        "brakes": String

        "concessionaireId": Concessionaire

    }

DELETE / buses /{id}

Params:

Id: bus ID

Response:

    {

        "message": String

    }

Device:

GET /device

Response:

[

{

    "id": Integer,

    "ip": String

    "busId": Buses

    "status": String

    "devideTypeId": DeviceType

}

]

GET / device /{id}

Params:

Id: device ID

Response:

    {

    "id": Integer,

    "ip": String

    "busId": Buses

    "status": String

    "devideTypeId": DeviceType

    }

POST / device

Body(json):

    {

    "ip": String

    "busId": Buses

    "status": String

    "devideTypeId": DeviceType

    }

Response:

    {

    "id": Integer,

    "ip": String

    "busId": Buses

    "status": String

    "devideTypeId": DeviceType

    }

PUT / device

Body(json):

{ "id": Integer,

    "ip": String

    "busId": Buses

    "status": String

    "devideTypeId": DeviceType

}

Response:

    {

    "id": Integer,

    "ip": String

    "busId": Buses

    "status": String

    "devideTypeId": DeviceType

    }

DELETE / device /{id}

Params:

Id: device ID

Response:

    {

        "message": String

    }

* 1. How can be a hierarchical access to enable the front-end for querying devices belonging to a specific bus?

R/ a hierarchical access is

[http://127.0.0.1:8080/api/busDevice/{id}](http://127.0.0.1:8080/api/busDevice/%7bid%7d)

Where id is the specific bus

GET / busDevice/{id}

Params:

Id: bus ID

Response:

[

{

    "id": Integer,

    "ip": String

    "busId": Buses

    "status": String

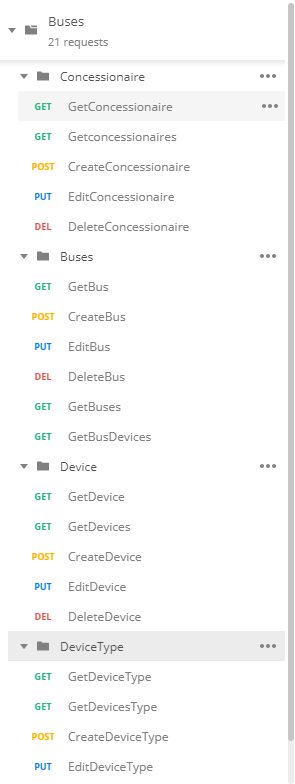
    "devideTypeId": DeviceType

}

]

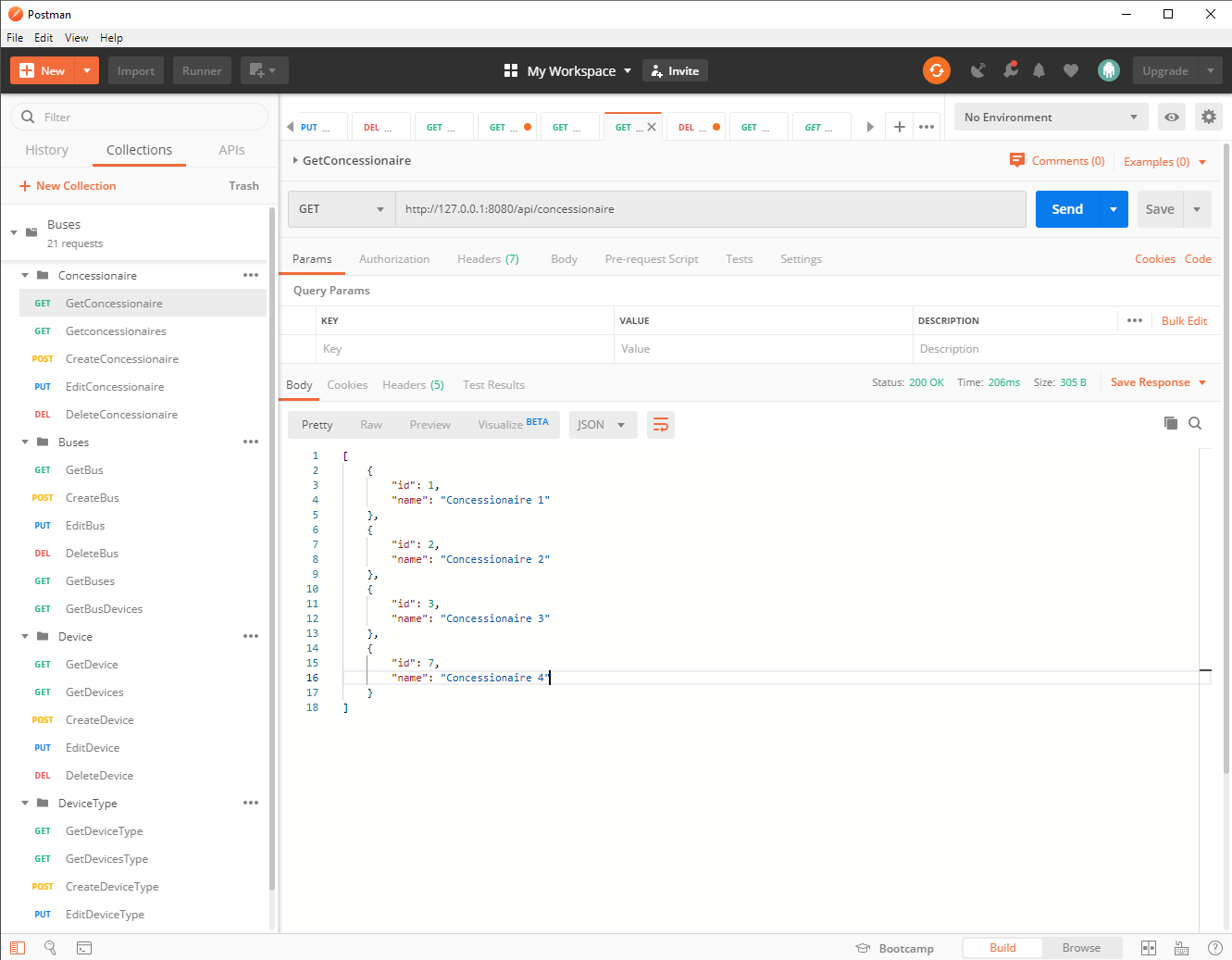
1. Test the API REST and attach the evidence. Postman is suggested.

I used postman for to test the API and I created these request

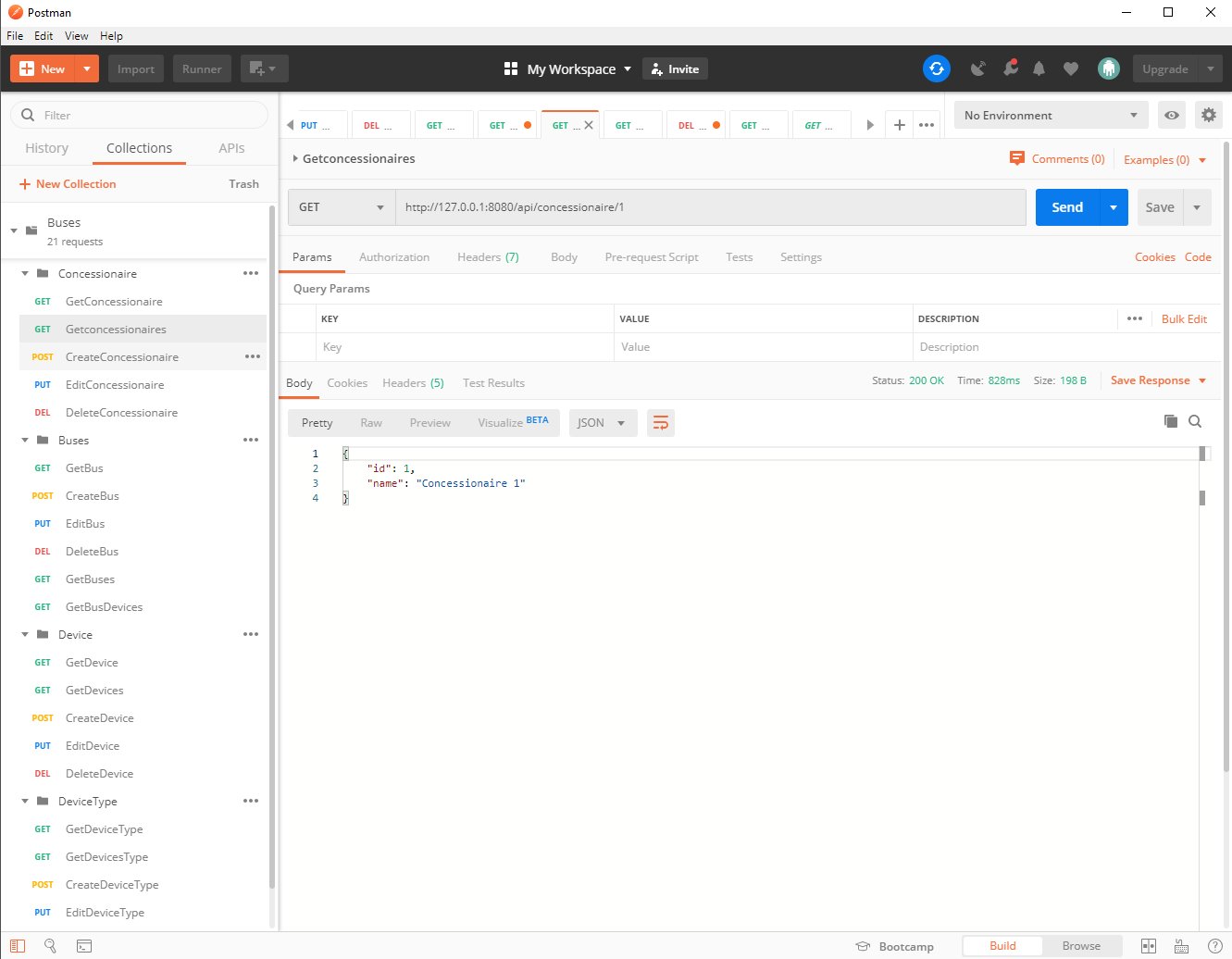


Concessionaire:

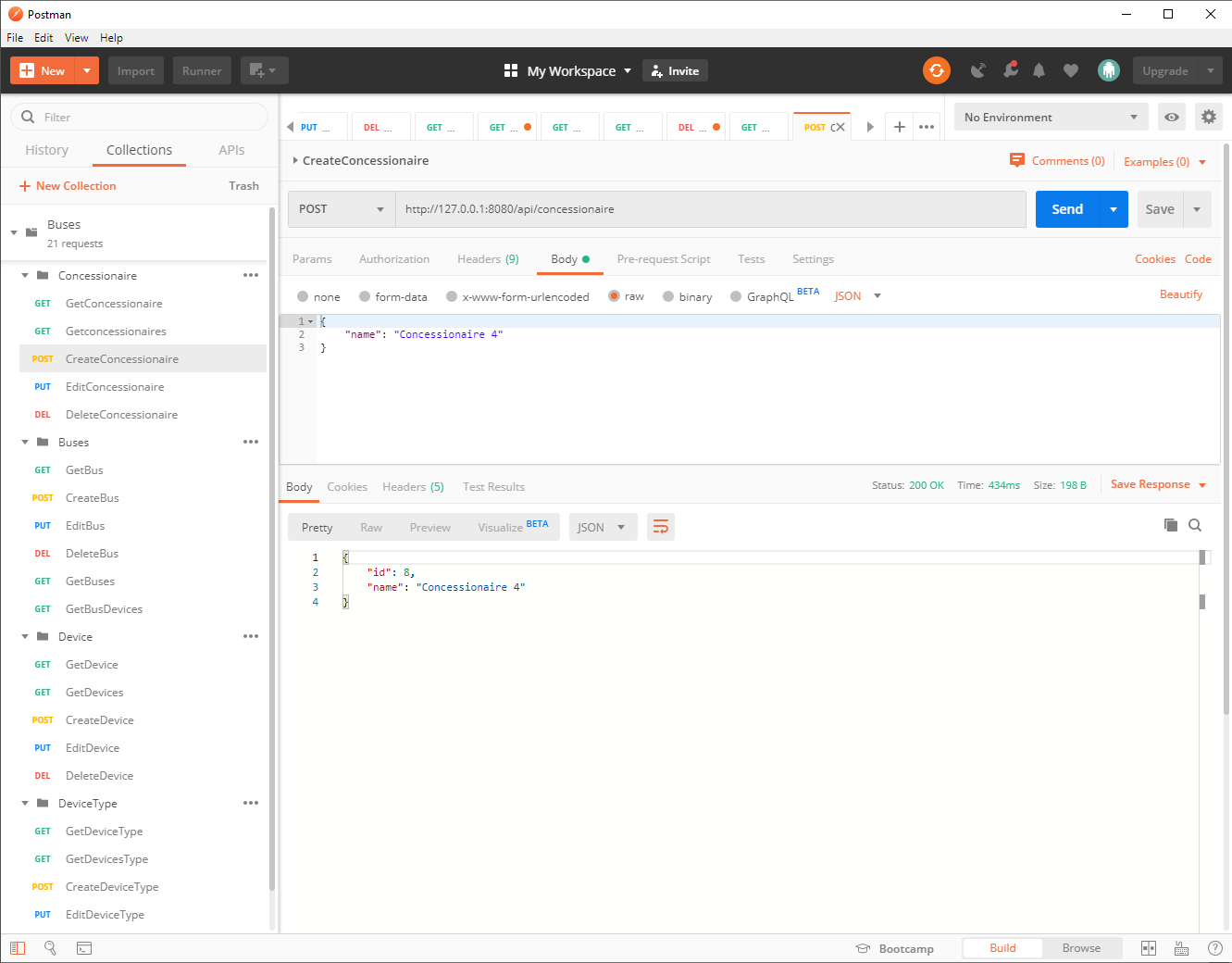
GET /concessionaire



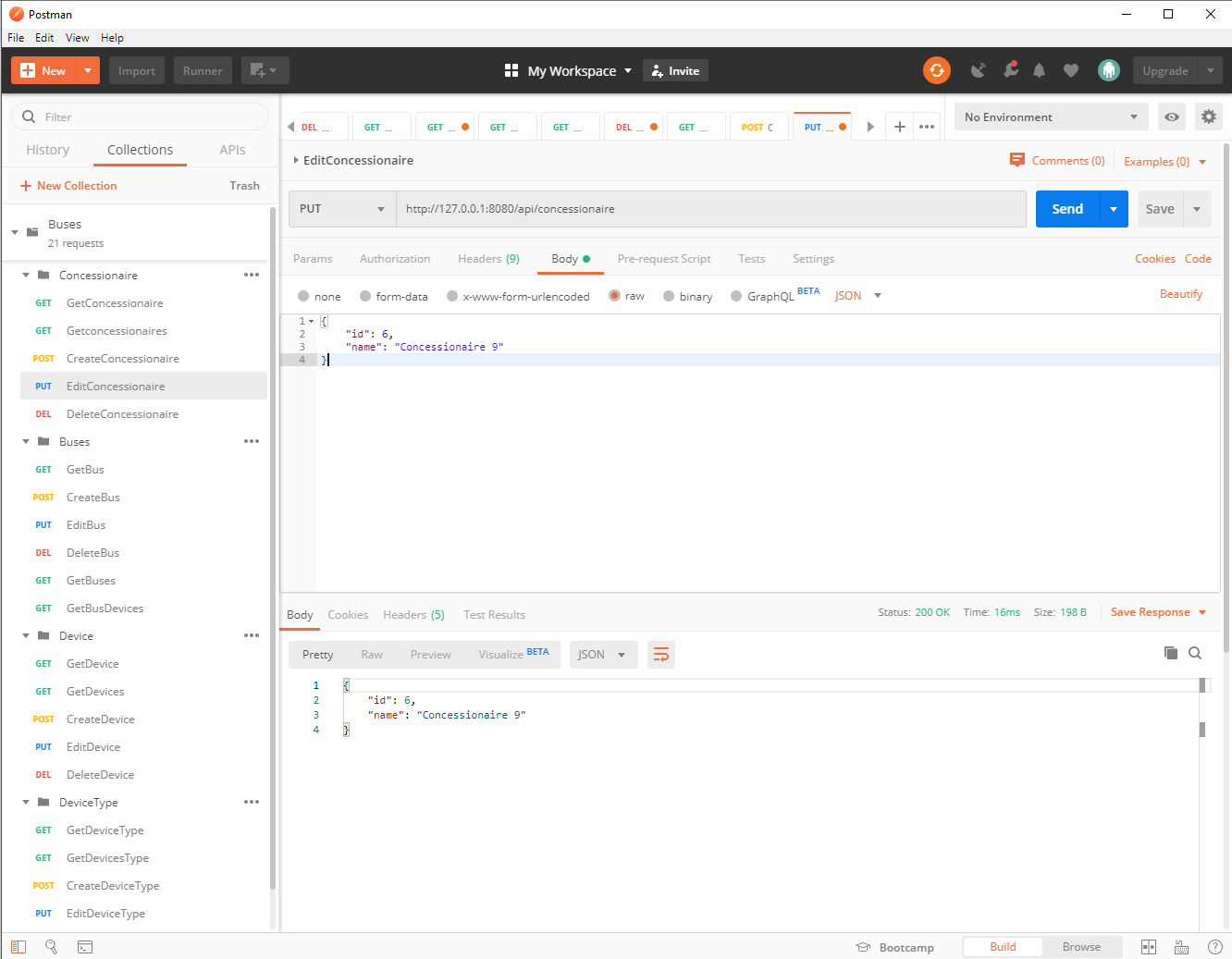
GET /concessionaire/{id}



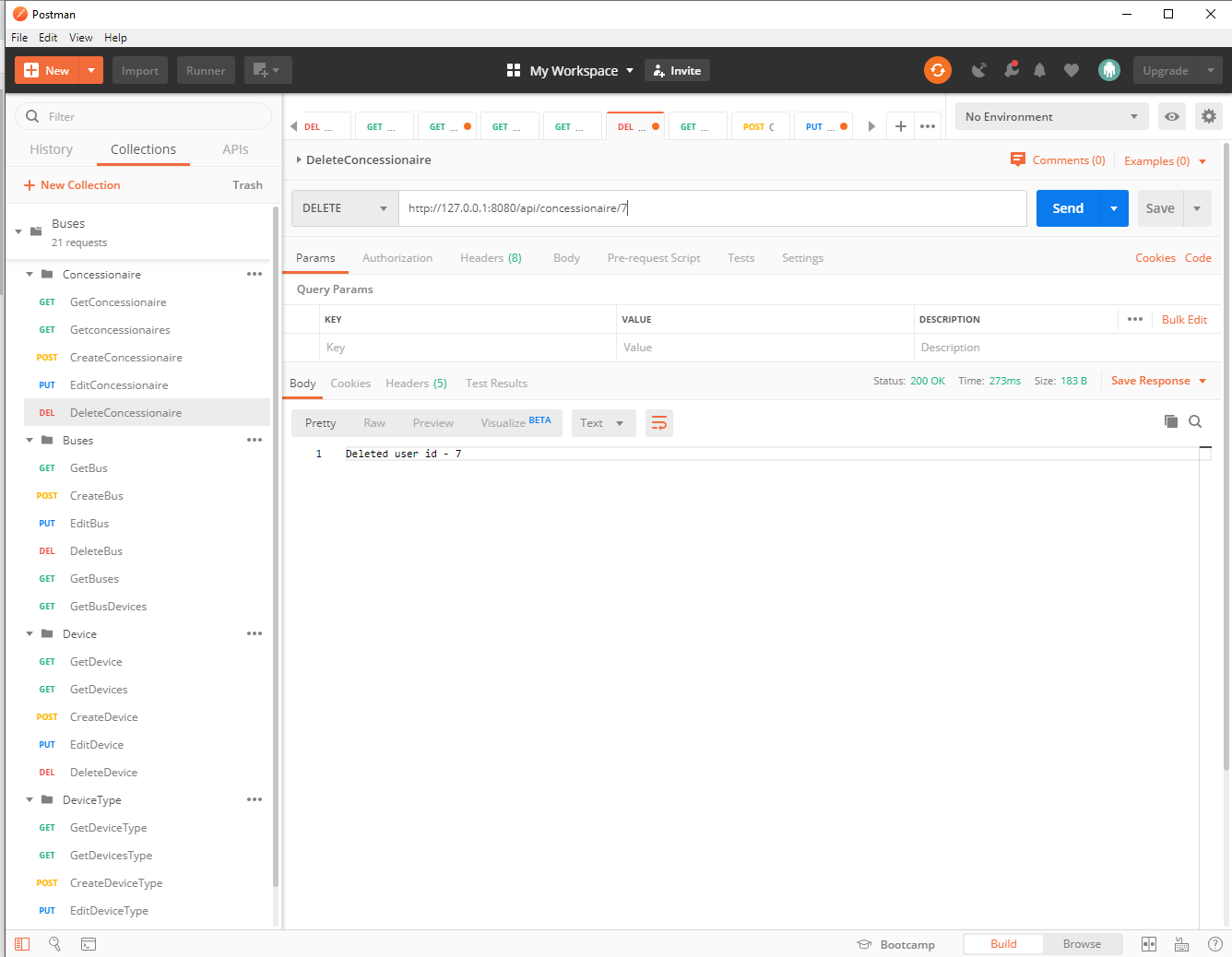
POST / concessionaire



PUT / concessionaire

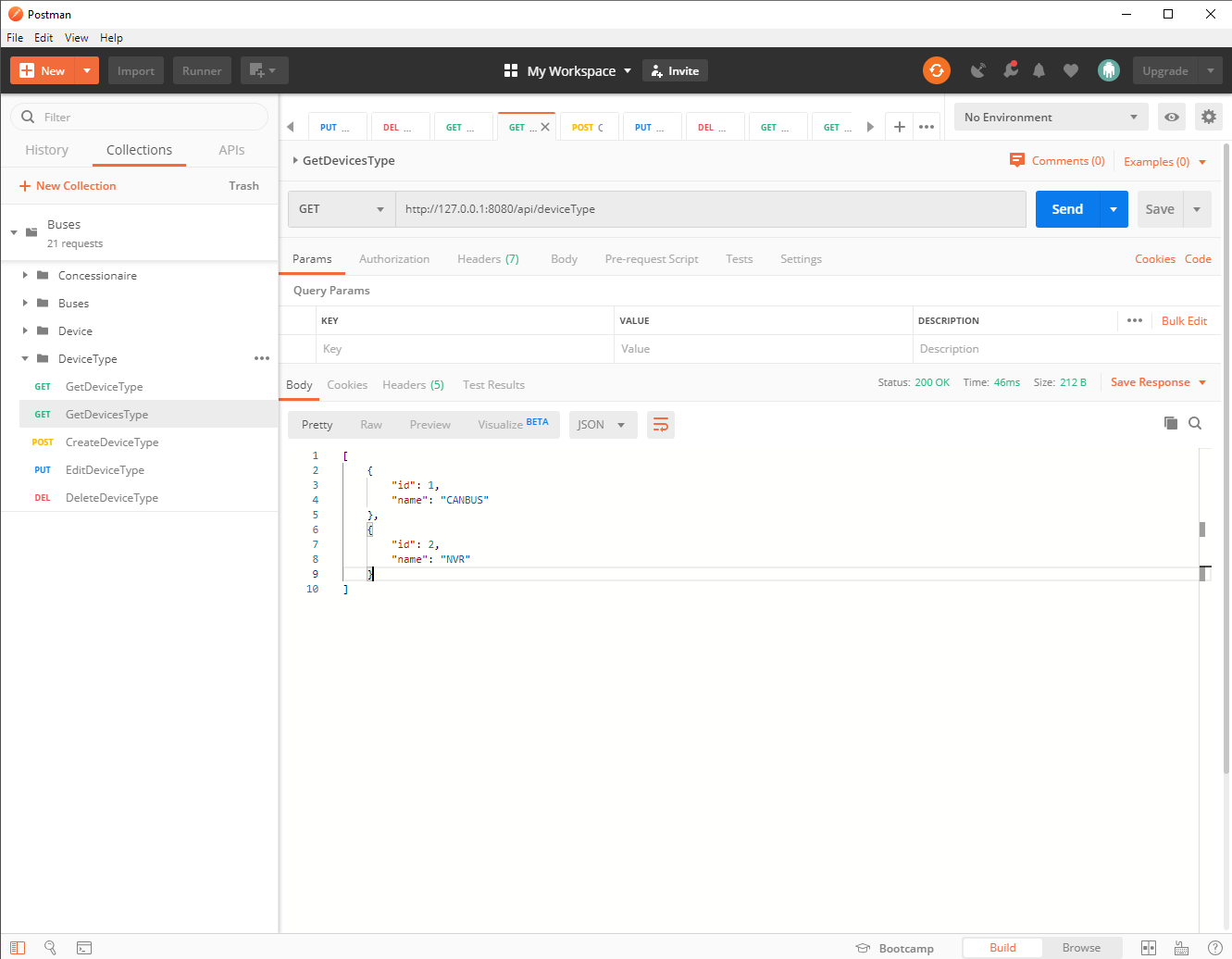


DELETE /concessionaire/{id}

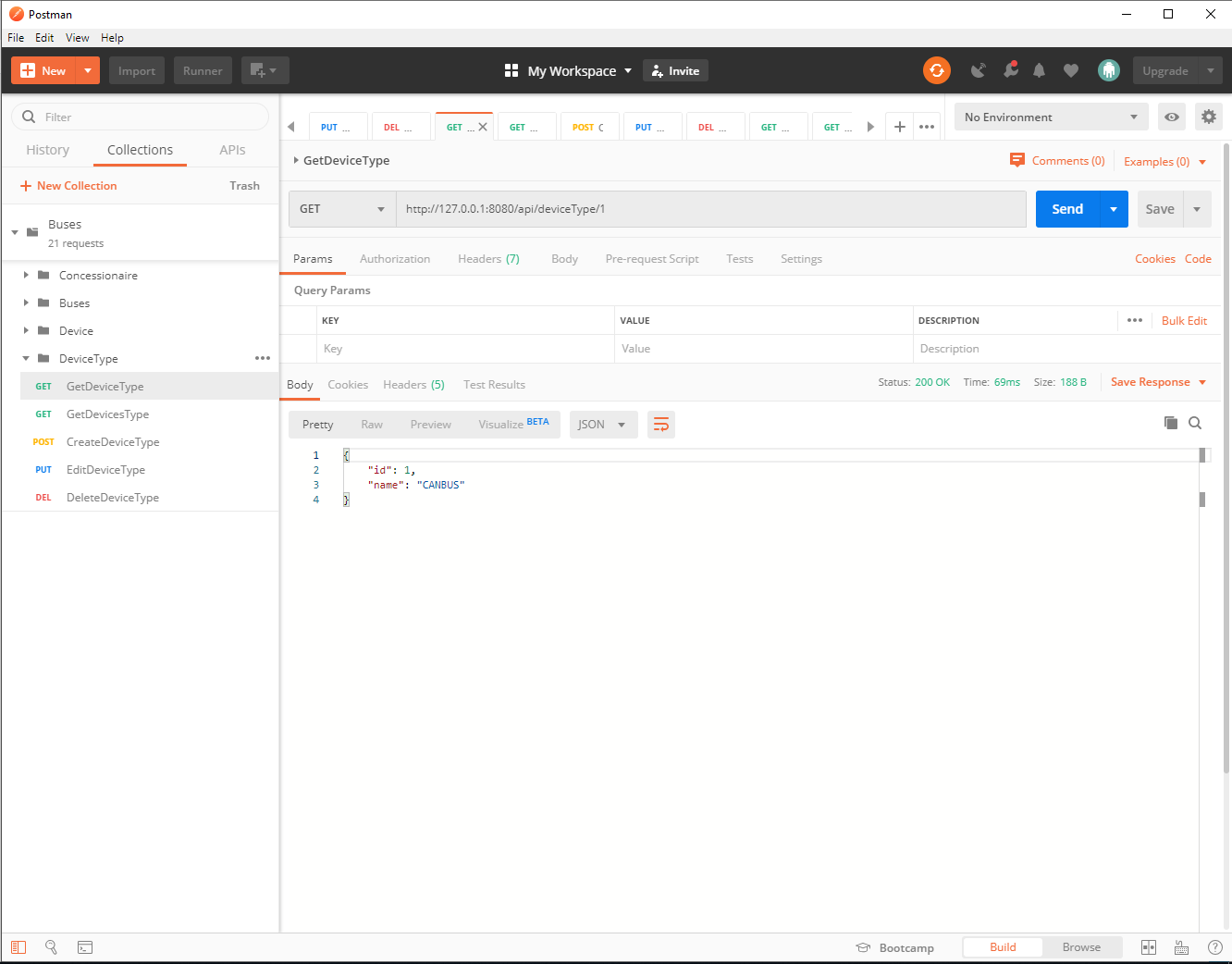


DeviceType:

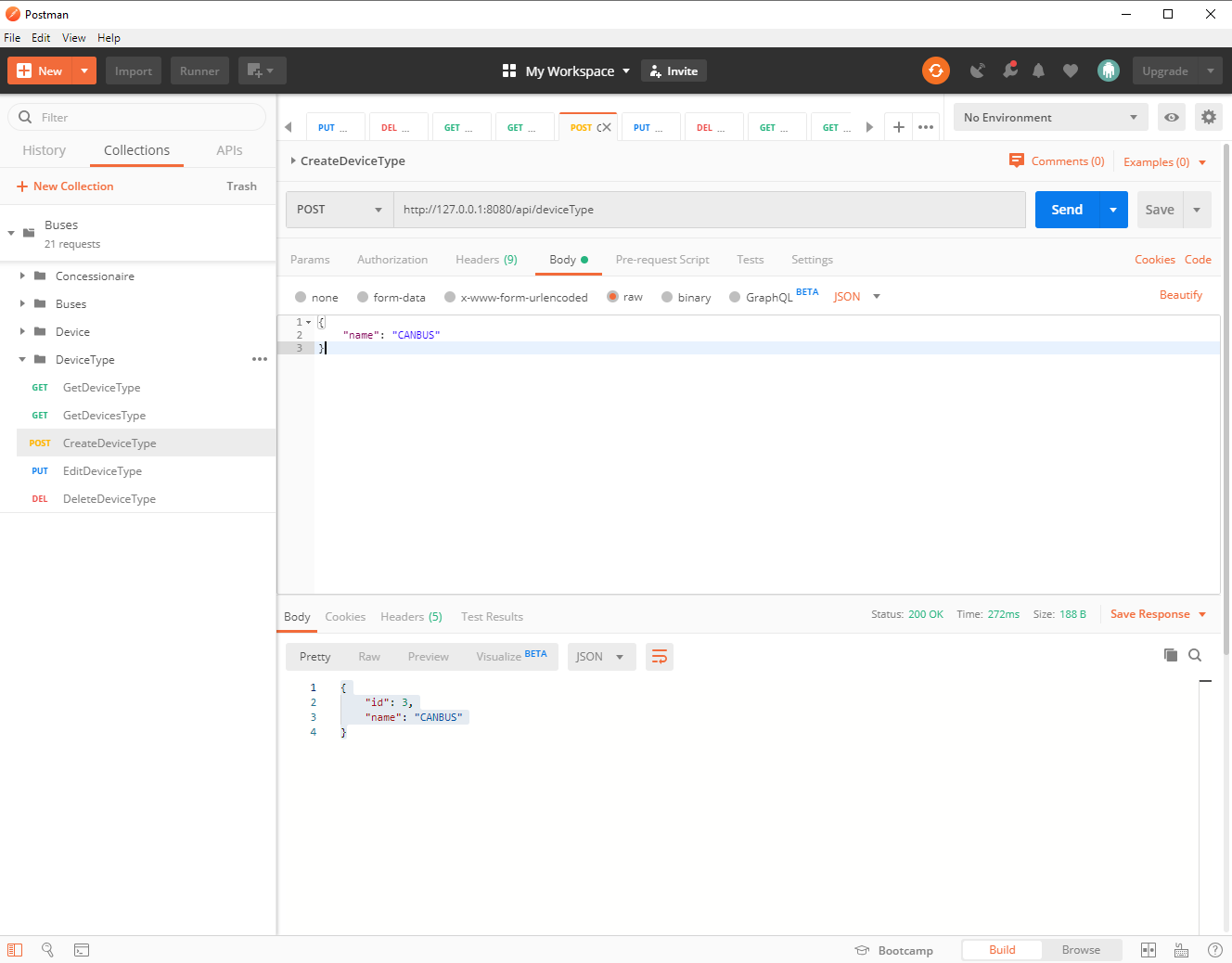
GET /deviceType



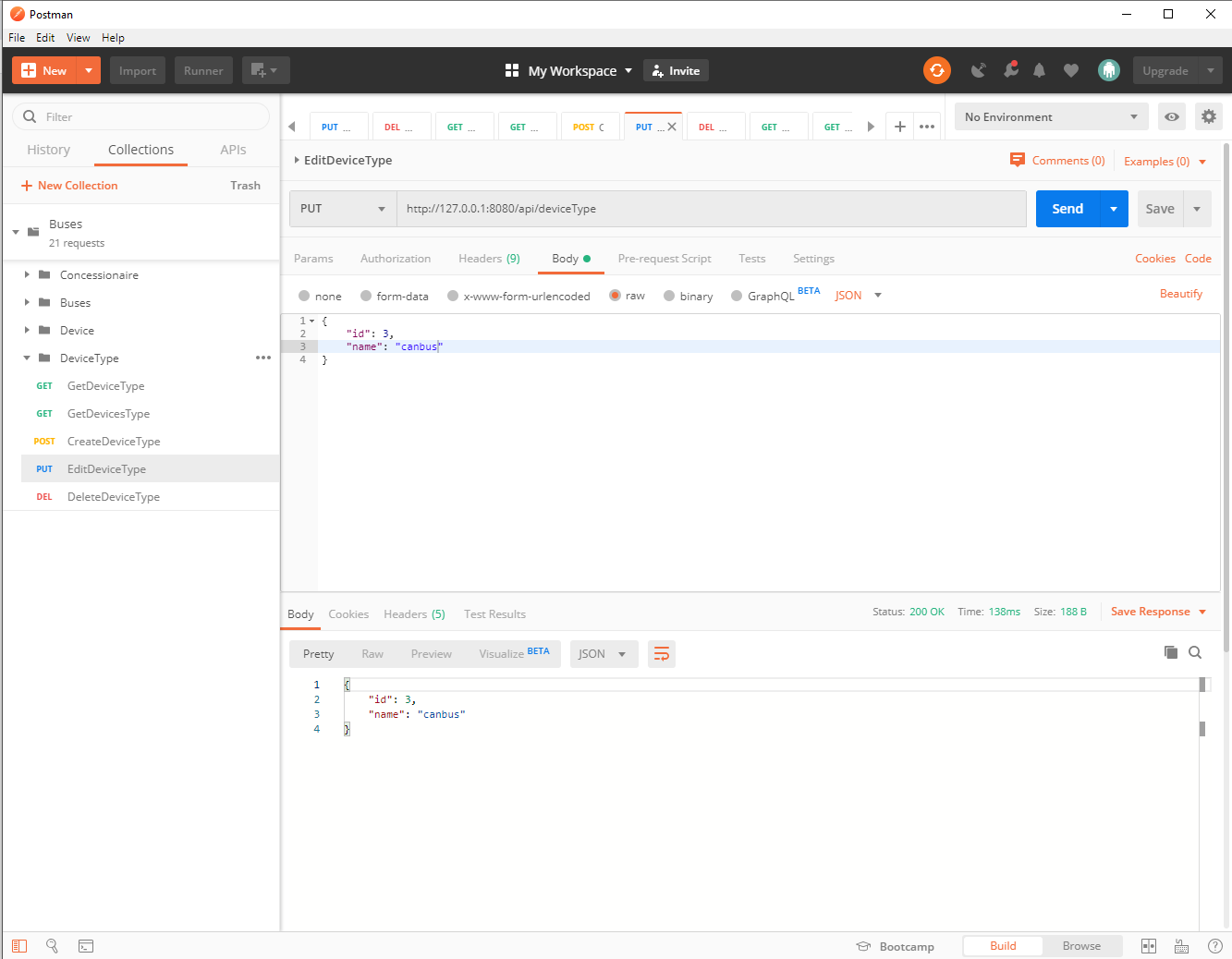
GET / deviceType /{id}



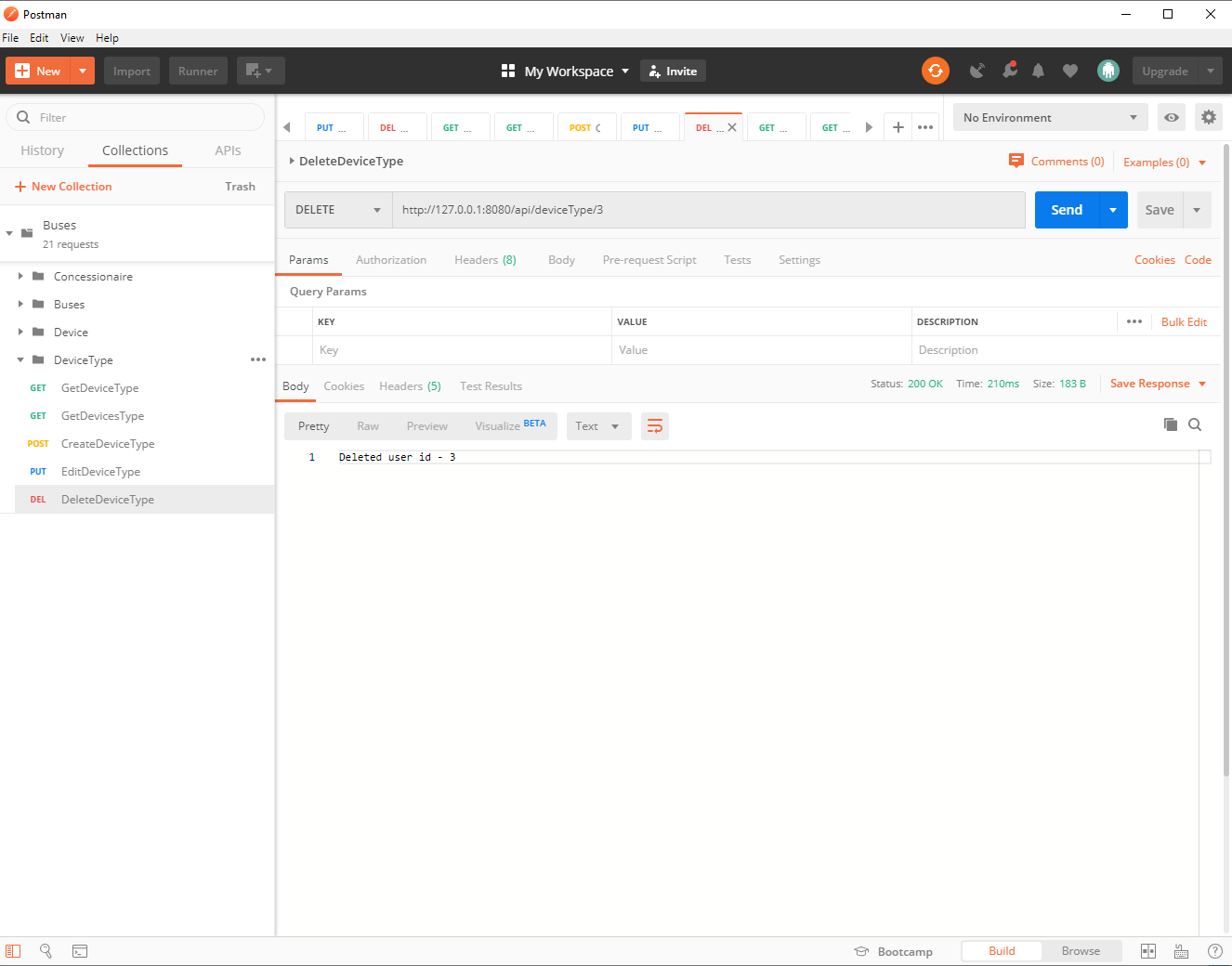
POST / deviceType



PUT / deviceType

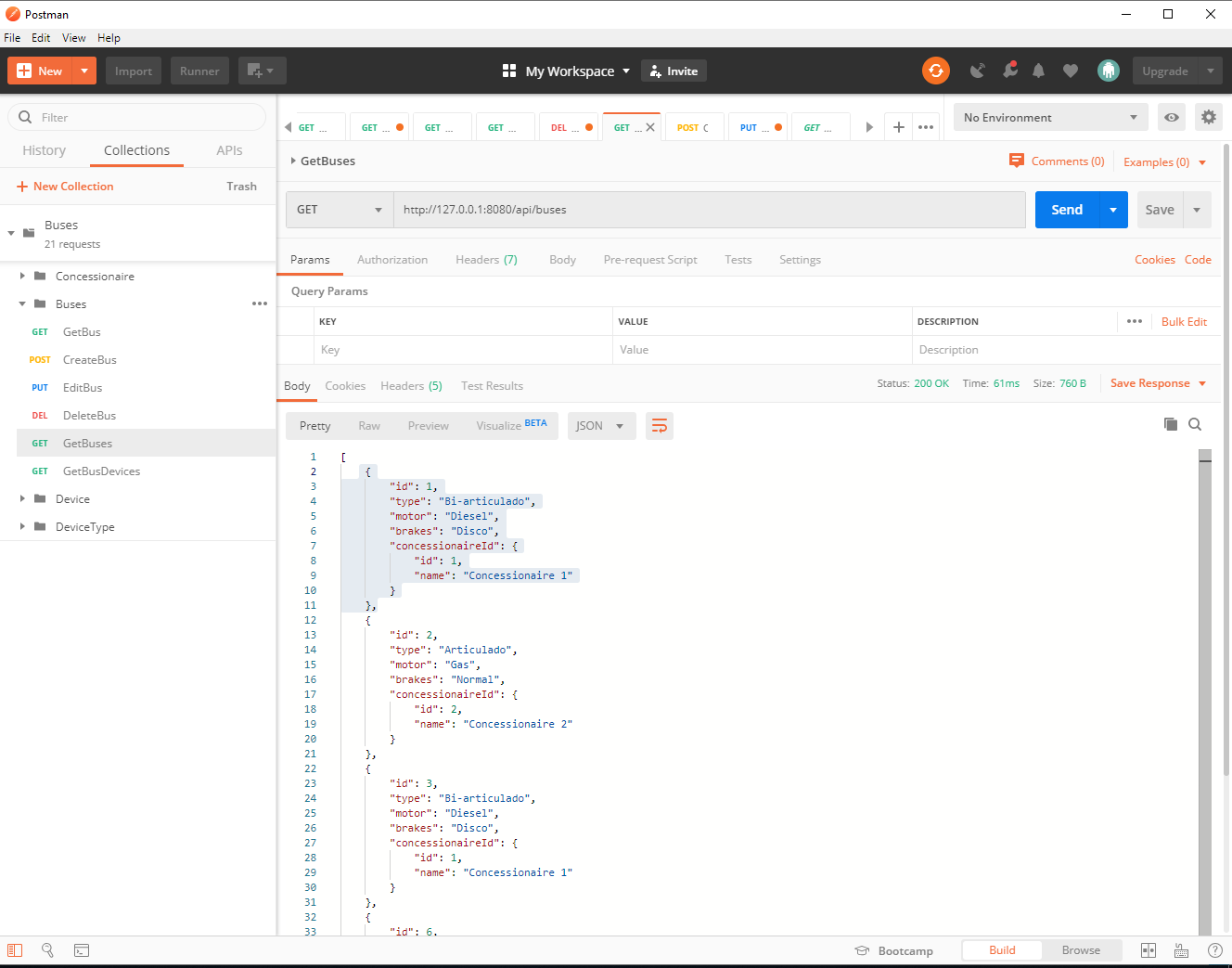


DELETE / deviceType /{id}

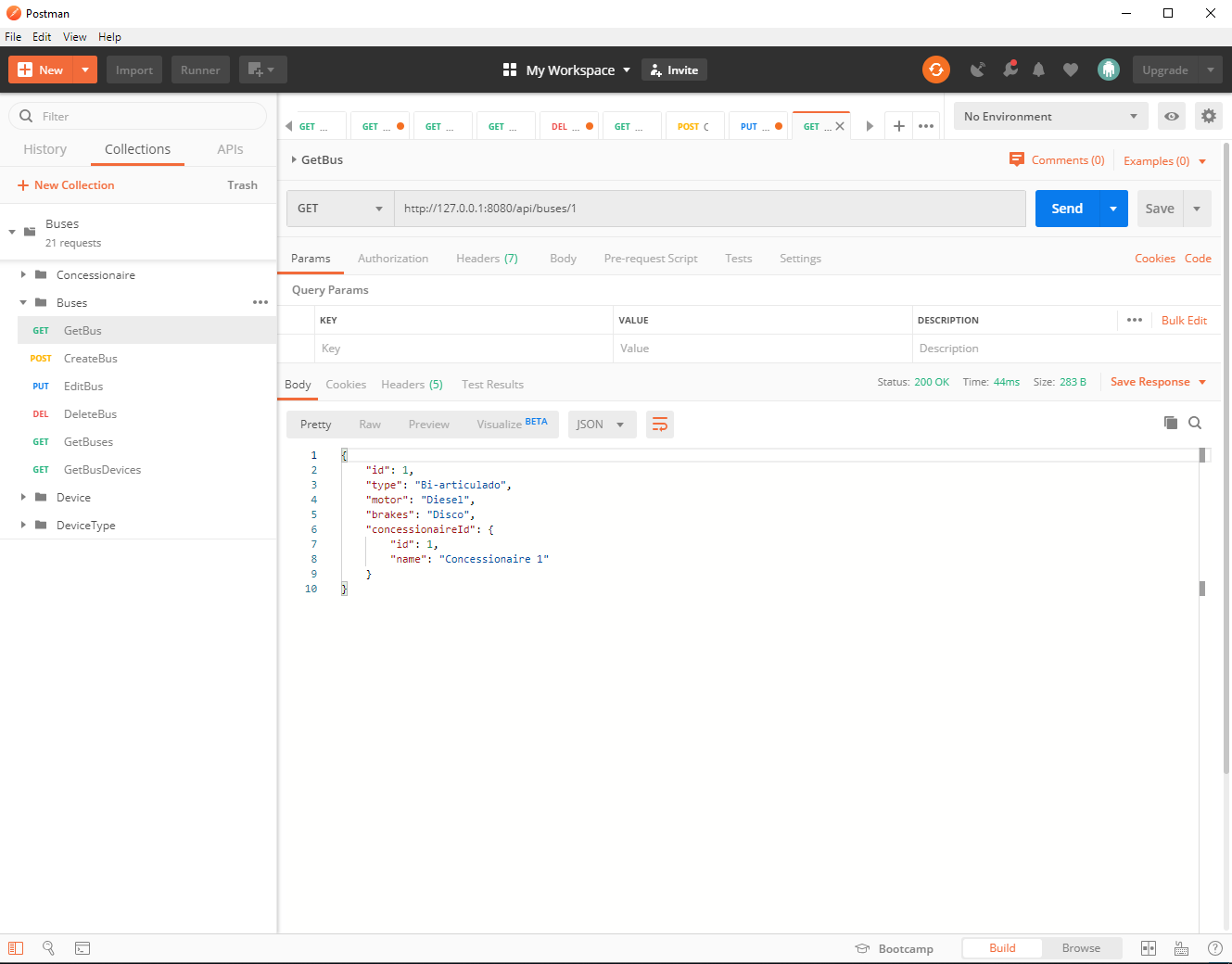


Buses:

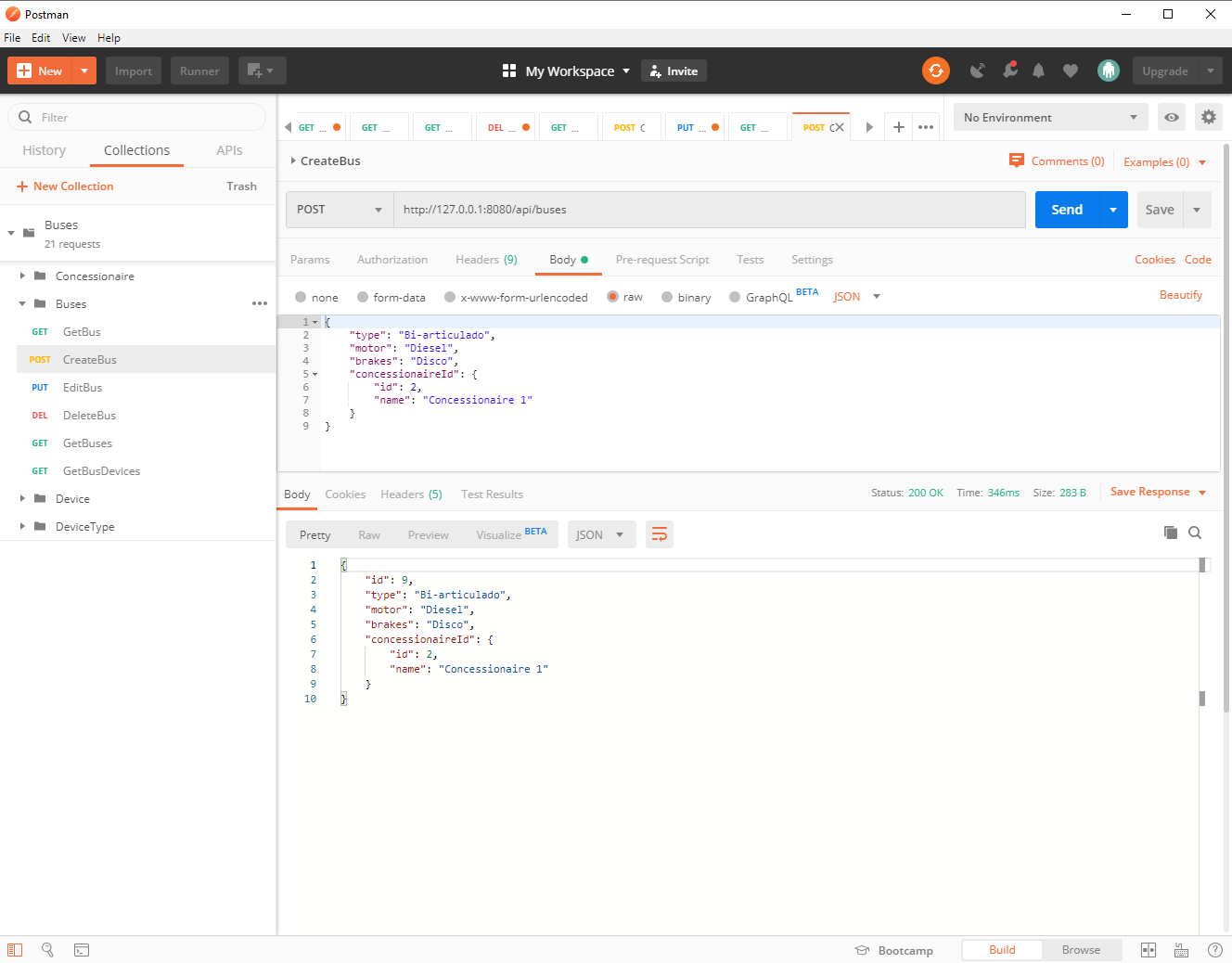
GET /buses



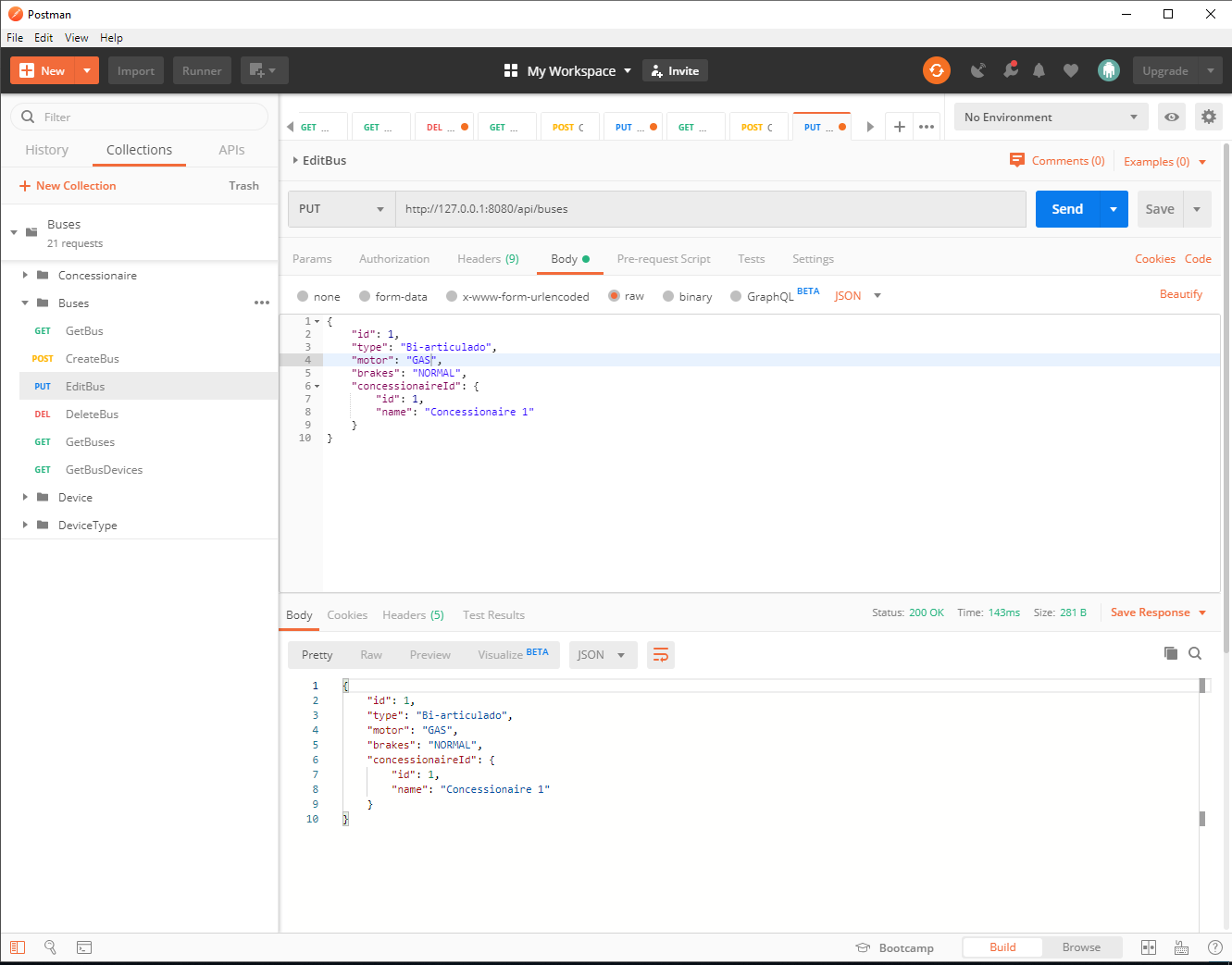
GET / buses /{id}



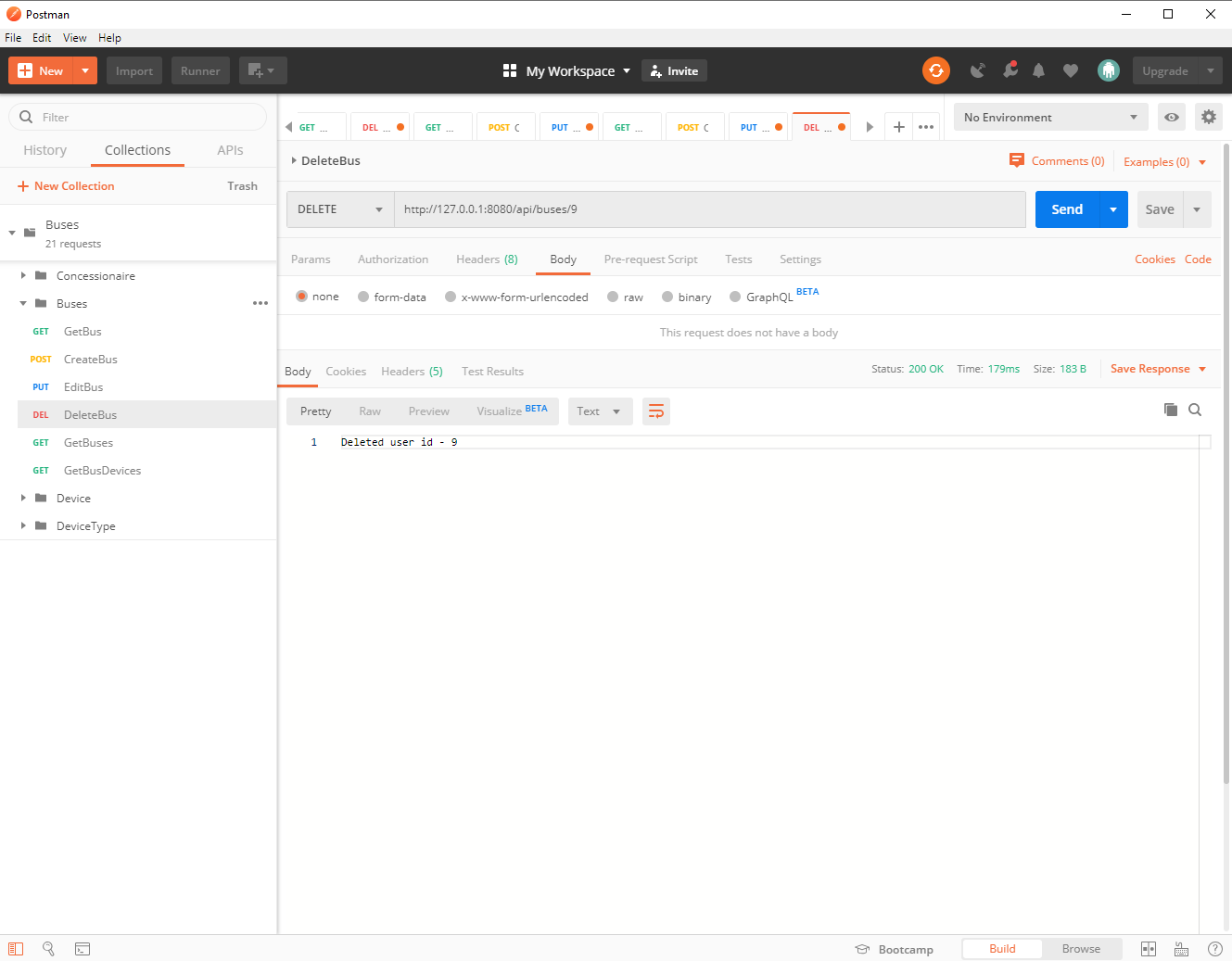
POST / buses



PUT / buses

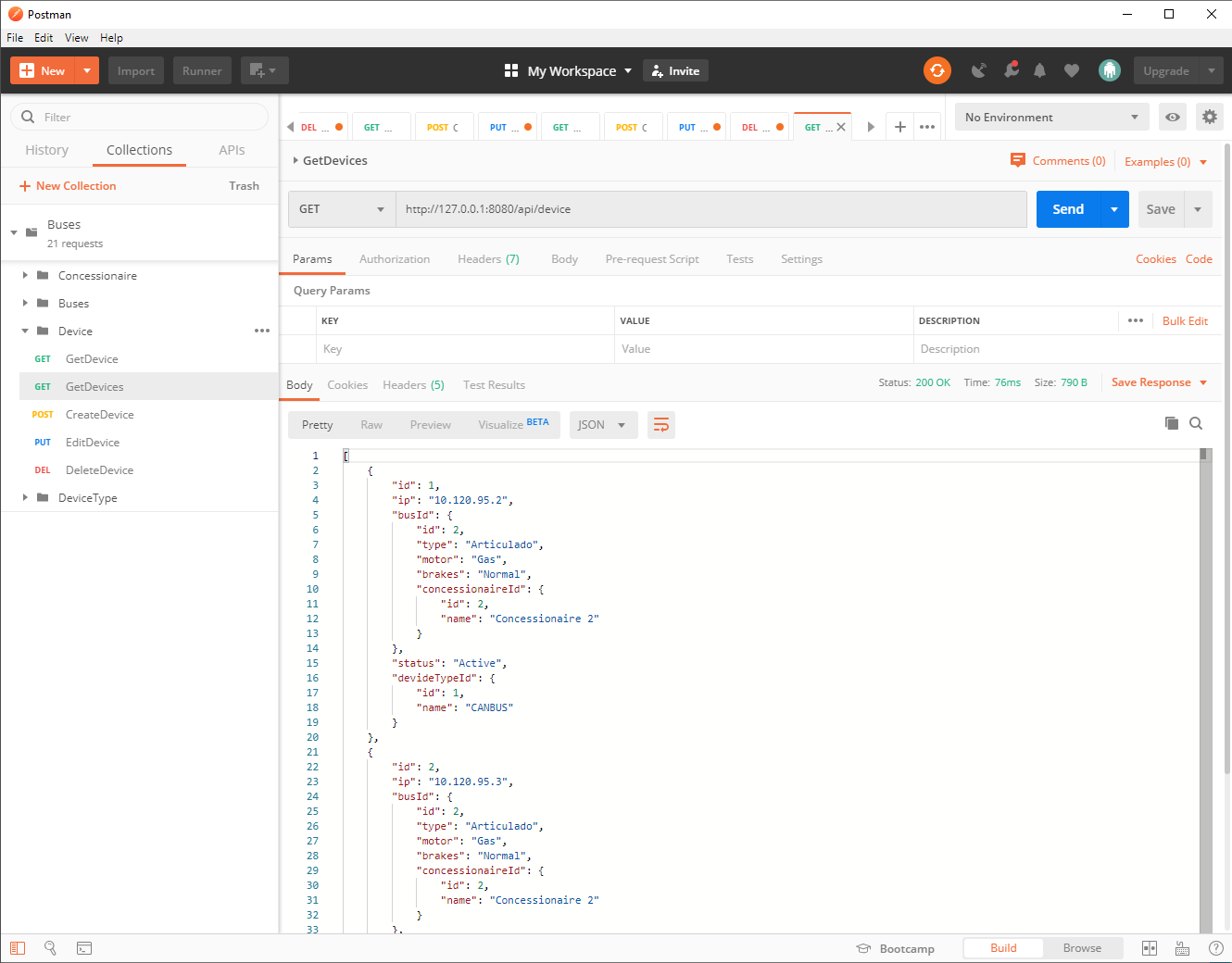


DELETE / buses /{id}

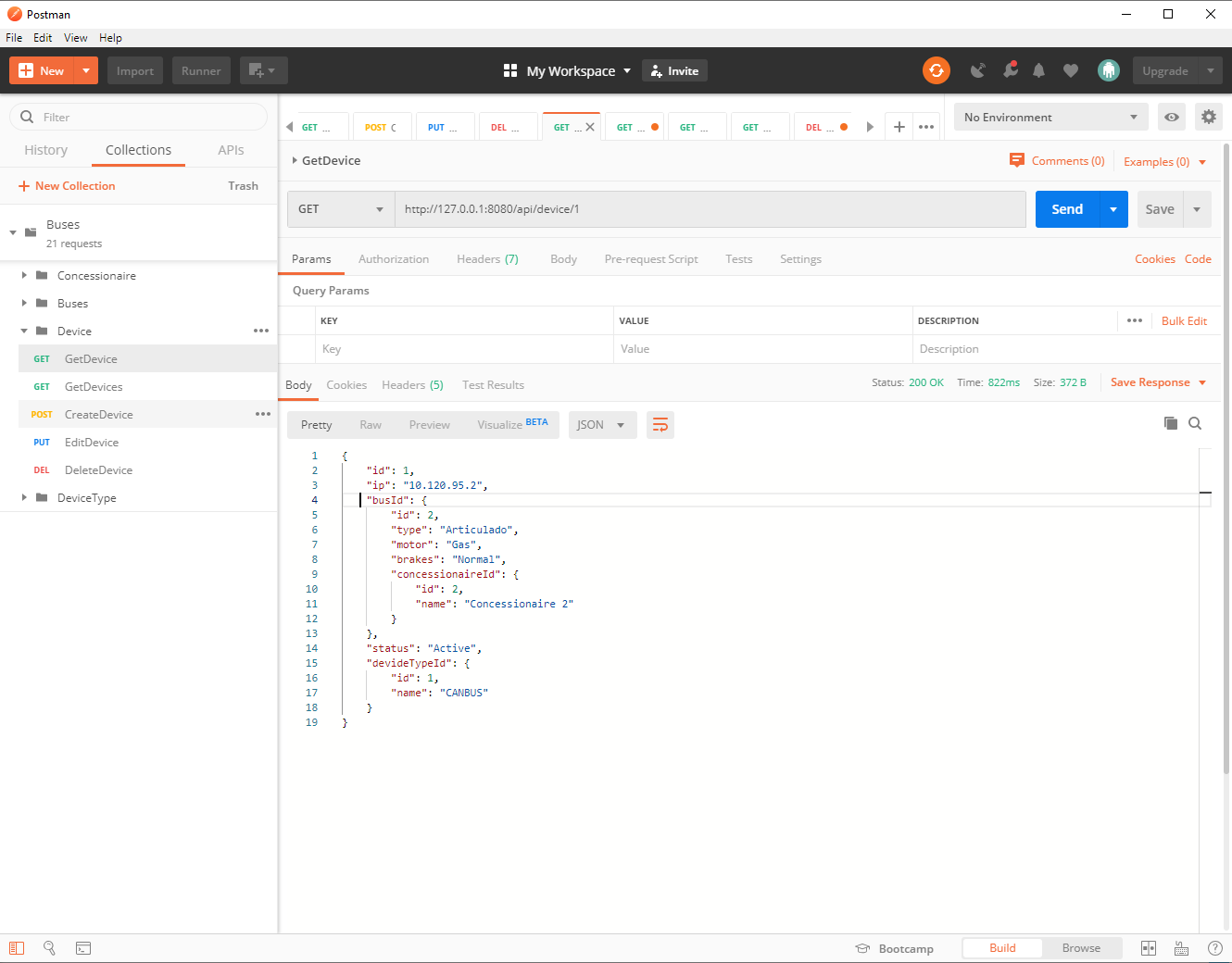


Device:

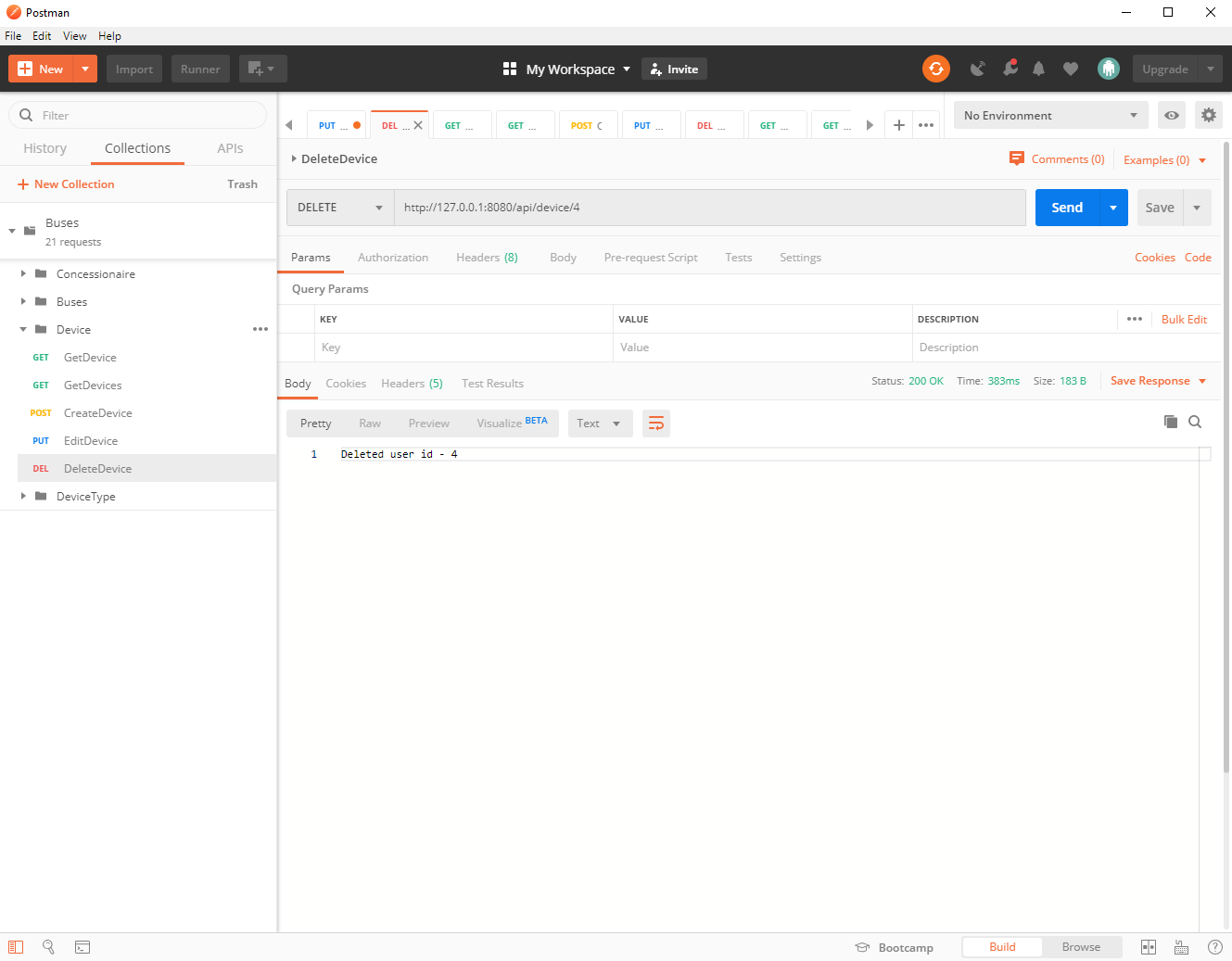
GET /device



GET / device /{id}



DELETE / device /{id}



GET /busDevice/{id}

