Smart Global Ecosystems - Guide to accessing data with a REST API

Guillermo Vega Gorgojo

April 2021 (updated in June 2021)

1. Introduction

This document is a guide for accessing some forestry datasets through a Web API. Specifically, the datasets are the following:

- The Spanish Forestry Inventory (IFN3)
- The Spanish Forestry Map (MFE50)

These datasets have been integrated and available for querying at this public endpoint: https://forestexplorer.gsic.uva.es/sparql/. A sample application that exploits this data is Forest Explorer.

The problem is that working with such integrated dataset requires knowledge of several technologies, at least the RDF data model and the SPARQL query language. Instead, you are going to use a REST API that I have configured for this case.

It is recommended some basic knowledge of the following topics (check the resources):

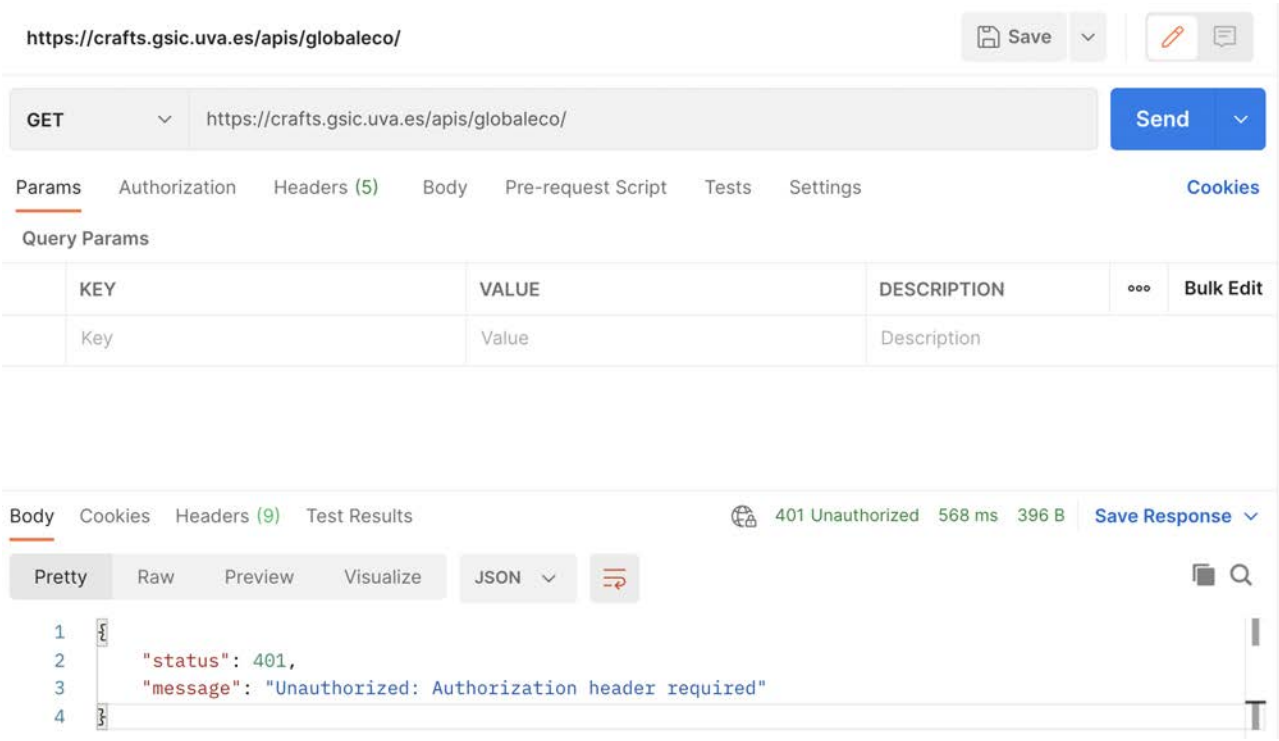
- REST APIs
  - Learn REST: A RESTful Tutorial, by Todd Fredrich
  - REST: a FAQ, by Diogo Lucas
  - RFC 2616
- JSON
  - RFC 8259
- URIs and IRIs
  - RFC 3986
  - RFC 3987

In order to use the proposed API you are going to use a GUI REST client, Postman. This client allows you to save both calls and responses, is free, is easy to configure, and works on both Mac and PC, as well as in modern web browsers such as Firefox or Chrome.

2. First steps with Postman and the API

1. Go to https://www.postman.com/downloads/ and get the Postman app or try the Web version (this is the one I will use)
2. Create an account and sign in when prompted
3. If this is your first time launching Postman, a welcome screen appears. Click **Create new** in order to create a new request
4. Insert the following URI into the box next to **GET**: `https://crafts.gsic.uva.es/apis/globaleco/`
5. Click **Send**

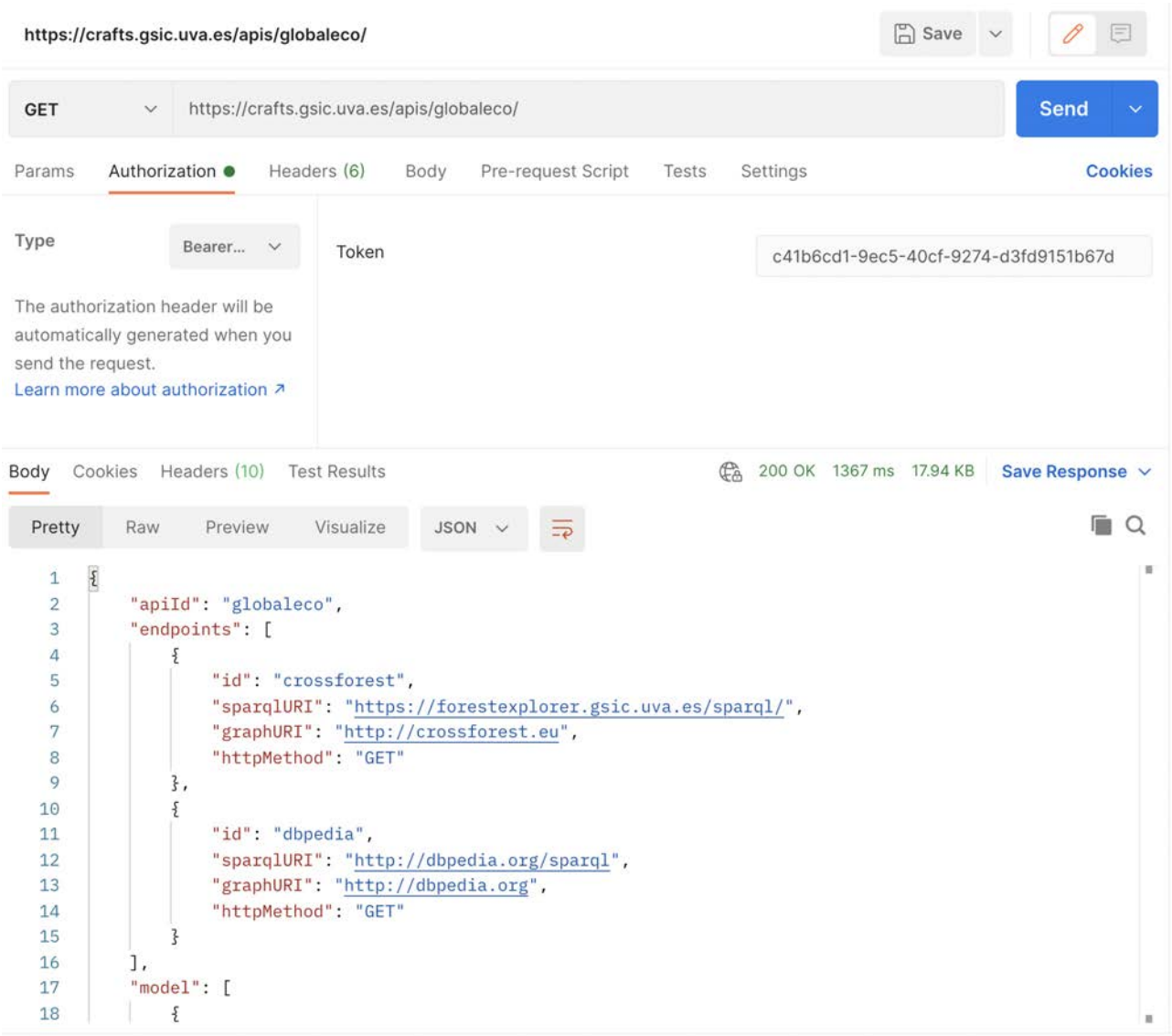
   The response appears in the lower pane. For example:



6. You are unauthorized to perform such operation!

   Click the **Authorization** tab, select **Bearer Token** as type, and then use `c41b6cd1-9ec5-40cf-9274-d3fd9151b67d` as **Token**.

   Click **Send** and you will get a valid response:



The response is the configuration file that I've prepared to set up the API.

3. A quick look to the configuration file

The configuration file is a long JSON object with the instructions to access the target datasets through a REST API. Don't worry, we'll just cover the basics to understand what is about and how you can use it to access the data.

The configuration file has the following keys:

- `apiId` : the name of the API (value `globaleco`)
- `endpoints` : an array with the information to access the data sources. The primary source is `crossforest` , containing the IFN3 and MFE50 datasets. `dbpedia` is a secondary source that is only employed to gather additional information about species
- `model` : an array with all the resource types exposed by the API. Each resource type includes:
  - An `id` such as `Tree` , `Position` , `Species` , etc.
  - Several attributes within the arrays `oprops` , `dprops` , and `types` . Here you should only care about the `label` (corresponding to the attribute name); the rest of information is employed for extracting the data from the endpoints
- `queryTemplates` : an array with a number of SPARQL query templates. I have prepared them to easily query the contents of the datasets without requiring knowledge of SPARQL. Each template includes:
  - An `id` such as `allSpecies` , `countTrees` , `treesInBox` , etc.
  - A textual `description` of the template (read it to grasp what is the template purposed for)
  - The actual `template` . This is for the query engine, so you don't need to read it

- The response of a query is essentially a table, the column names correspond to the `variables`
- A template can be parametrized providing values to the declared `parameters`. Note that parameters can be optional and have an expected type

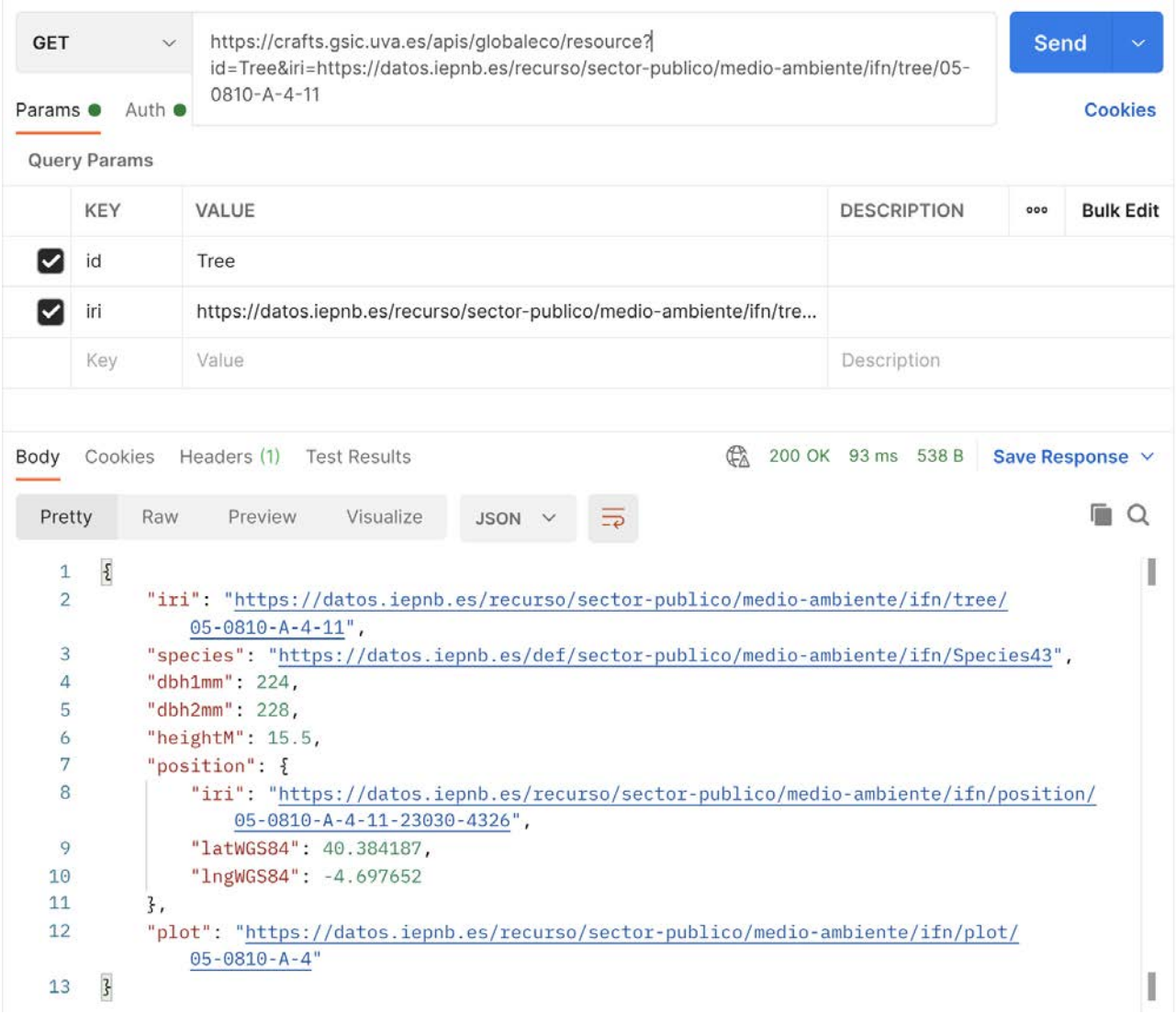
4. Using the API to retrieve representations of resources

It is very easy to get a representation of a resource with a known IRI. This just requires a GET operation with this format: `https://crafts.gsic.uva.es/apis/globaleco/resource?id={id}&iri={iri}`

You only have to replace `{id}` with the resource type and `{iri}` with the IRI of the resource. Let's try with an example, we have a tree with IRI `https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tree/05-0810-A-4-11` :

1. Check the configuration file `https://crafts.gsic.uva.es/apis/globaleco/` in Postman to identify the id of the resource type: `Tree`
2. Craft the URI of the GET operation following the format above:

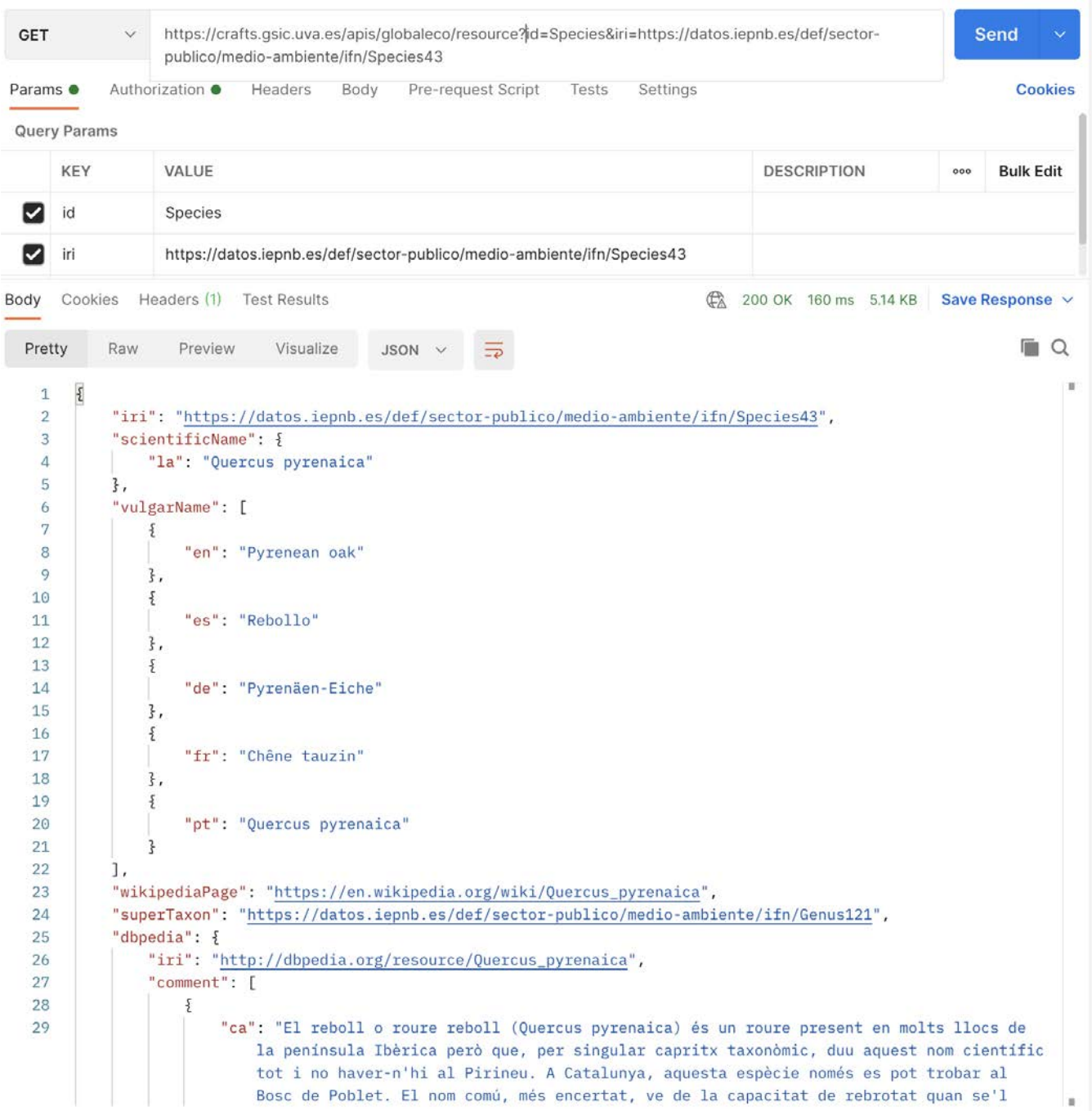
   `https://crafts.gsic.uva.es/apis/globaleco/resource?id=Tree&iri=https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tree/05-0810-A-4-11`
3. Click **Create new** in Postman to create a new request
4. Insert the crafted URI into the box next to **GET**
5. Click the **Authorization** tab, select **Bearer Token** as type, and then use `c41b6cd1-9ec5-40cf-9274-d3fd9151b67d` as **Token**
6. Click **Send** to get your response:



The response is a JSON object with the same keys defined for a `Tree` in the model of the API.

You are probably wondering what species is `https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43`. You can easily find it with a new GET request (when crafting the new URI check its resource type in the API model to set the right `id` parameter, it is `Species` in this case):

`https://crafts.gsic.uva.es/apis/globaleco/resource?id=Species&iri=https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43`



Similarly, you can retrieve information about the plot with IRI `https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/plot/05-0810-A-4` . The corresponding URI is:

`https://crafts.gsic.uva.es/apis/globaleco/resource?id=Plot&iri=https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/plot/05-0810-A-4`

5. Retrieve multiple representations of resources with one call

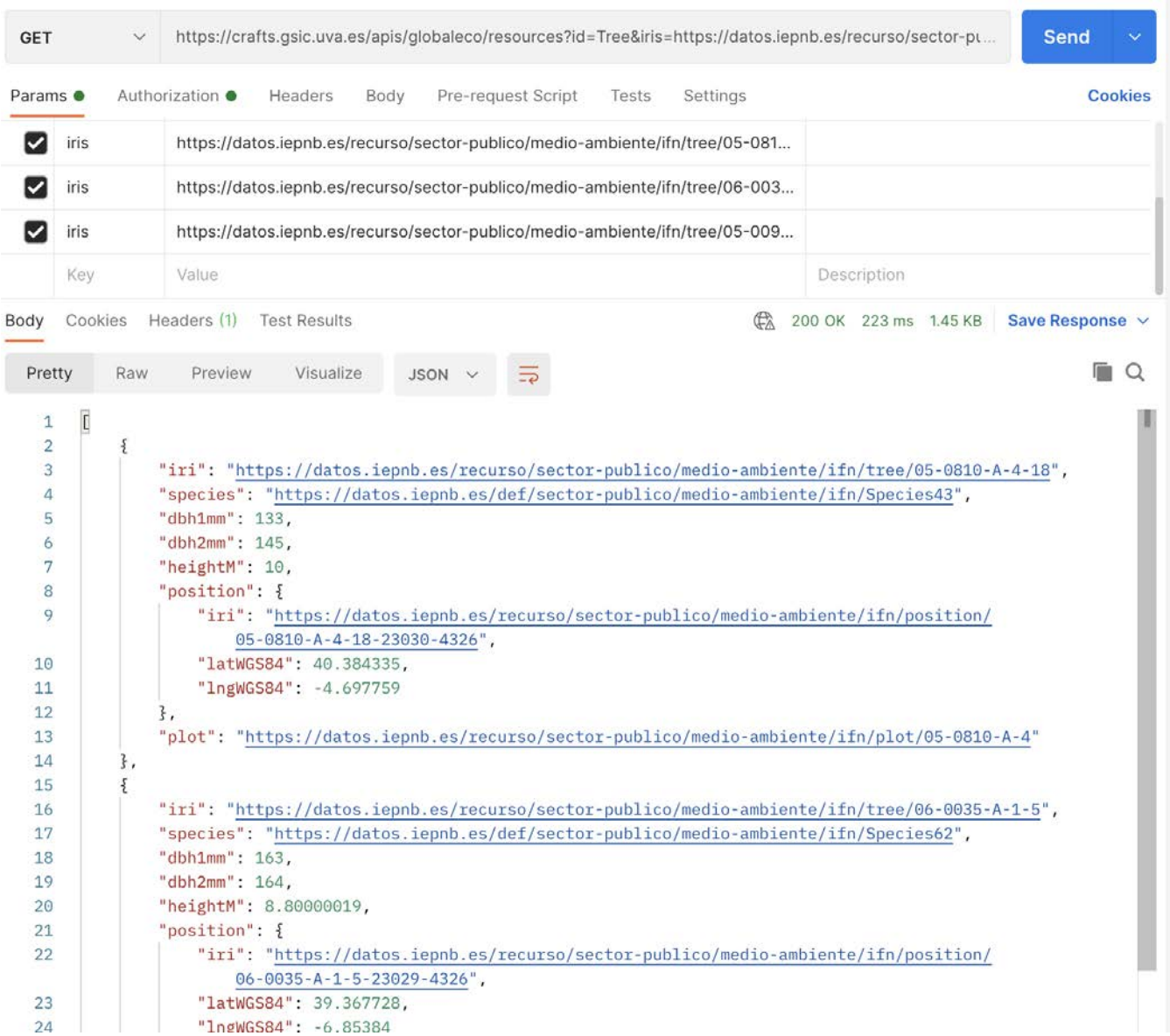
Imagine that you have many resources of the same type, e.g. trees `https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tree/05-0810-A-4-18` , `https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tree/06-0035-A-1-5` , and `https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tree/05-0093-A-3C-6` .

You can retrieve all their representations with just one call. The format of the corresponding GET operation is:

`https://crafts.gsic.uva.es/apis/globaleco/resources?id={id}&iris={iriA}&iris={iriB}...`

In our example:

```
https://crafts.gsic.uva.es/apis/globaleco/resources?
id=Tree&iris=https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tree/05-0810-A-4-
18&iris=https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tree/06-0035-A-1-
5&iris=https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tree/05-0093-A-3C-6
```



6. How can I know the IRIs of resources?

We have seen how we can use the API to retrieve representations of resources. However, this requires knowing somehow their IRIs in advance. Query templates are intended to fulfil this need. As you can see in the config file, I have prepared a number of query templates. Let's examine how to work with them.

First of all, this is the format of the GET operation for query templates:

```
https://crafts.gsic.uva.es/apis/globaleco/query?id={id}&{parA}={valA}&{parB}={valB}...
```

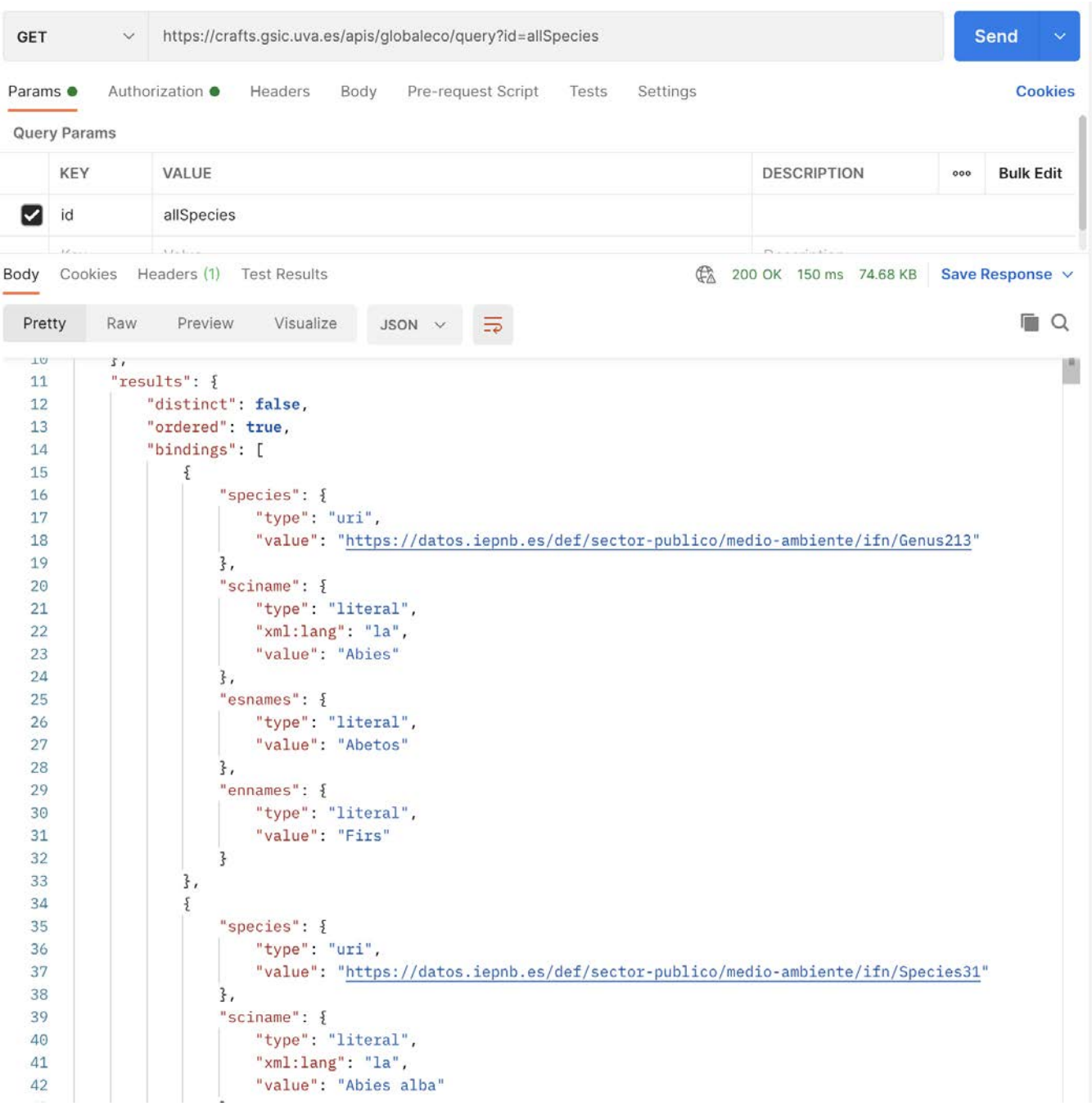
Note that the `id` can be found in the config file, while suitable parameters and values are set for the query template at hand. Let's try with the query template `allSpecies` (check the config file):

```
{
  "id": "allSpecies",
  "description": "Obtain all the species (variable \"species\") with their scientific names (variable
  "template": "select ?species ?sciname (group_concat(distinct ?esname;separator=\"; \") as ?esnames)
  "variables": [
    "species",
```

```
      "sciname",
      "esnames",
      "ennames"
    ],
    "parameters": [],
    "endpoint": "crossforest"
  }
```

In this case there are no query parameters, so you only need to make this GET call:

```
https://crafts.gsic.uva.es/apis/globaleco/query?id=allSpecies
```



We get the answer directly from the SPARQL endpoint in JSON-LD format. It is a regular JSON object, although a bit more verbose than needed. Anyway, the important thing to look is the `bindings` array. Each object in the array is a valid answer (a row of the table of results) with the same fields defined in the query template. You can browse the list of species and get the IRIs of the ones you are interested, e.g. `https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43` is the IRI of *Quercus pyrenaica*.

7. Setting parameters in queries

We are now prepared to pose more complex queries. Let's go with the query template `countTrees` (check the config file):


```
{
   "id": "countTrees",
   "description": "Obtain the count of the trees (variable \"count\") of an optional species (paramet
   "template": "SELECT count(?tree) as ?count\nWHERE {\n  ?tree a <https://datos.iepnb.es/def/sector-
   "variables": [
      "count"
   ],
   "parameters": [
     {
       "label": "species",
       "type": "iri",
       "optional": true
     },
     {
       "label": "lngwest",
       "type": "number",
       "optional": true
     },
     {
       "label": "lngeast",
       "type": "number",
       "optional": true
     },
     {
       "label": "latnorth",
       "type": "number",
       "optional": true
     },
     {
       "label": "latsouth",
       "type": "number",
       "optional": true
     }
   ],
   "endpoint": "crossforest"
}
```


Since all the parameters are optional, we can just count all the trees in the dataset by making this GET call:

`https://crafts.gsic.uva.es/apis/globaleco/query?id=countTrees`

In a next step we can get the count of all the trees of species *Quercus pyrenaica* by setting the parameter `species` to `https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43` :

`https://crafts.gsic.uva.es/apis/globaleco/query?id=countTrees&species=https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43`

We can browse the list of species and find that the IRI of genus *Quercus* is `https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Genus121` . We can thus ask how many *Quercus* we have:

`https://crafts.gsic.uva.es/apis/globaleco/query?id=countTrees&species=https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Genus121`

We can also specify a specific region of interes, so we can have:

```
GET    https://crafts.gsic.uva.es/apis/globaleco/query
    id    countTrees
    species    https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Genus121
    lngwest    0
    lngeast    3
```

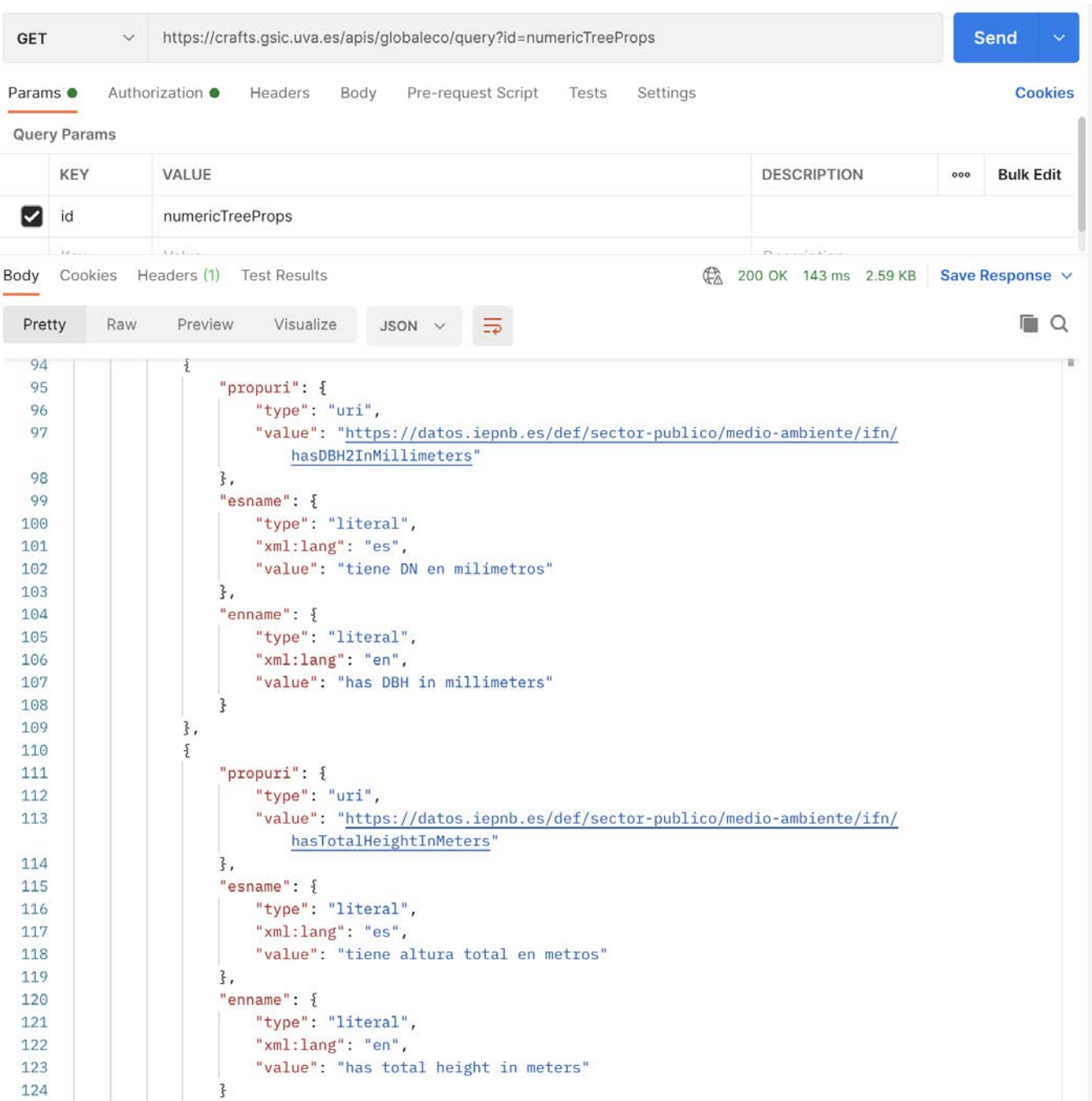
```
latnorth    40
latsouth    37
```

And the resulting URI is `https://crafts.gsic.uva.es/apis/globaleco/query?id=countTrees&lngwest=0&lngeast=3&latnorth=40&latsouth=37&species=https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Genus121` (note that the order of query parameters is not relevant).

8. More complex queries

Let's continue with more query templates: `numericTreeProps` is intended to obtain the IRIs of some tree properties to apply in the query templates `maxPropTrees` and `avgPropTrees`. As `numericTreeProps` has no parameters, you can get the results by calling:

```
https://crafts.gsic.uva.es/apis/globaleco/query?id=numericTreeProps
```



Note that there are seven different properties available for querying about numeric properties:

```
https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/hasDBH1InMillimeters
https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/hasDBH2InMillimeters
```

```
https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/hasTotalHeightInMeters
https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/hasVolumeWithBarkInM3
https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/hasVolumeWithoutBarkInM3
https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/hasVolumeFirewoodInM3
https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/hasVolumeIncreaseInM3
```

We are ready to use query templates `maxPropTrees` and `avgPropTrees` (note that they require a `propiri` parameter). After checking these query templates in the API config it should be easy to employ them. For example:

```
GET    https://crafts.gsic.uva.es/apis/globaleco/query
    id    avgPropTrees
    propiri    https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/hasTotalHeightInMeters
    species    https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43
    lngwest    0
    lngeast    3
    latnorth    40
    latsouth    37
```

We can also get the trees (and their locations) in a bounding box (setting an optional `species` as parameter) by using the query template `treesInBox`:


```
    {
      "id": "treesInBox",
      "description": "Obtain trees (variable \"tree\") of an optional species (parameter \"species\"
      "template": "SELECT DISTINCT ?tree ?lat ?lng\nWHERE {\n  ?tree a <https://datos.iepnb.es/def/se
      "variables": [
        "tree",
        "lat",
        "lng"
      ],
      "parameters": [
        {
          "label": "species",
          "type": "iri",
          "optional": true
        },
        {
          "label": "lngwest",
          "type": "number",
          "optional": true
        },
        {
          "label": "lngeast",
          "type": "number",
          "optional": true
        },
        {
          "label": "latnorth",
          "type": "number",
          "optional": true
        },
        {
          "label": "latsouth",
          "type": "number",
          "optional": true
        },
        {
          "label": "limit",
          "type": "integer",
          "optional": true
        },
        {
          "label": "offset",
          "type": "integer",
          "optional": true
```

```
        }
      ],
      "endpoint": "crossforest"
    }
```

This query template works as expected, but take into account that its results can be paginated using parameters `limit` and `offset`. For example, this query gets the first ten trees of species *Quercus pyrenaica* in the same bounding box as before:

```
GET    https://crafts.gsic.uva.es/apis/globaleco/query
    id    treesInBox
    species    https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43
    lngwest    0
    lngeast    3
    latnorth    40
    latsouth    37
    limit    10
    offset    0
```

The resulting URI is thus `https://crafts.gsic.uva.es/apis/globaleco/query?id=treesInBox&lngwest=0&lngeast=3&latnorth=40&latsouth=37&limit=10&offset=0&species=https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43`

If we want to get the following page of ten trees, then we have:

```
GET    https://crafts.gsic.uva.es/apis/globaleco/query
    id    treesInBox
    species    https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43
    lngwest    0
    lngeast    3
    latnorth     40
    latsouth     37
    limit     10
    offset     10
```

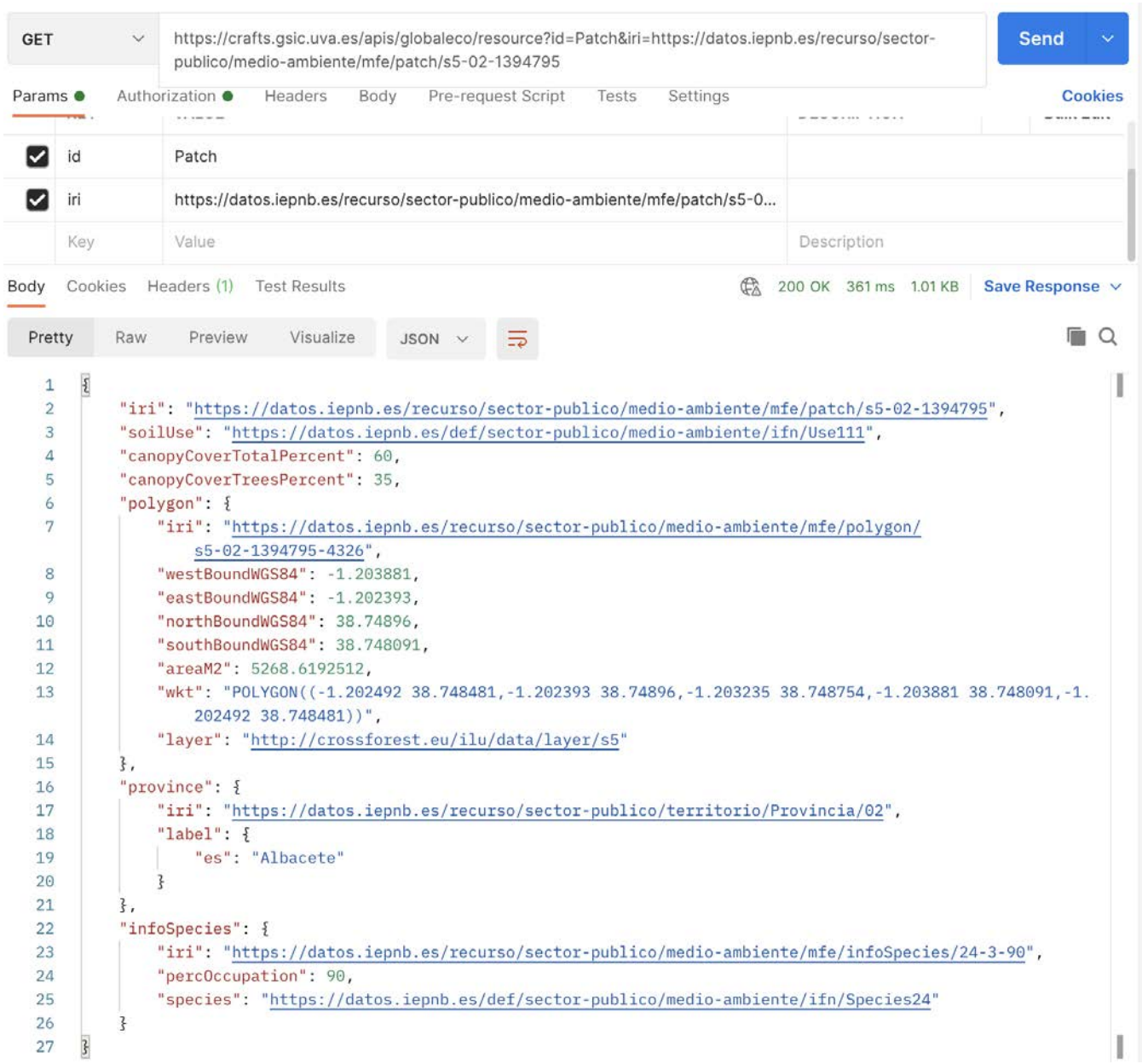
And the new URI for this query is `https://crafts.gsic.uva.es/apis/globaleco/query?id=treesInBox&lngwest=0&lngeast=3&latnorth=40&latsouth=37&limit=10&offset=10&species=https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43`

9. Remaining queries

The following query template is `plotsInBox`. This works very similar to the previous `treesInBox`, as the associated geometries of plots are points, as in the case of trees.

With respect to patches, their geometries are more complex, typically a polygon, defined by a sequence of points. For instance, we can easily obtain the representation of a patch if we know its IRI. In the case of patch `https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/mfe/patch/s5-02-1394795` we can make this call, as before: `https://crafts.gsic.uva.es/apis/globaleco/resource?`

id=Patch&iri=https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/mfe/patch/s5-02-1394795



Several things to observe:

- Patches have a soil use
- The polygon object has all the information about the patch geometry: bounds, area, layer, and sequence of points (check the API model)
- There is additional information about the species in the patch, canopy cover, and so on

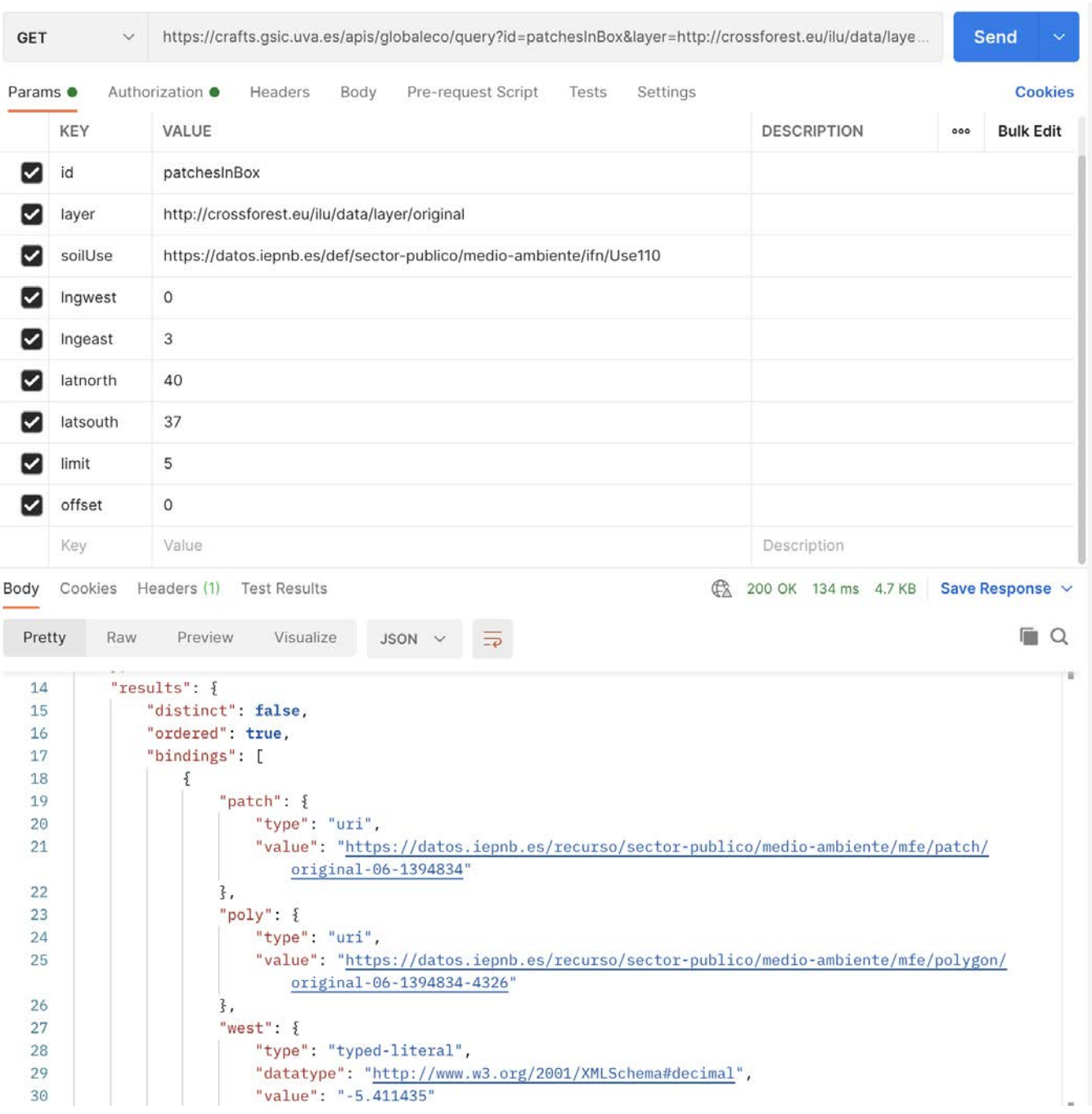
The template query `patchesInBox` is prepared for obtaining the patches in a bounding box (please check it in the config file of the API):

- You can set the bounding box parameters (as in the case of `plotsInBox` and `treesInBox`). Since the geometry of a patch is a polygon and not a point, the query will find the patches that are contained or intersect with the bounding box
- The pagination works exactly as before
- You can optionally set `areamin` as the minimum area (in square meters) for retrieving a patch
- You can optionally specify the `soilUse` of interest. Use the template query `allSoilUses` to find the IRIs of the soil uses available
- You can optionally specify the `layer` of interest. Note that there are three different layers of patches available (check them with the template query `allPatchLayers`)

We are now ready to use this template query. Let's try it to find the first five patches in the original layer, with wooded area as soil use, in the previous bounding box, and with a minimum area of 10000 square meters:

```
GET    https://crafts.gsic.uva.es/apis/globaleco/query
    id    patchesInBox
    layer    http://crossforest.eu/ilu/data/layer/original
    soilUse    https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Use110
    lngwest    0
    lngeast    3
    latnorth    40
    latsouth    37
    limit    5
    offset    0
```

And the new URI for this query is `https://crafts.gsic.uva.es/apis/globaleco/query?id=patchesInBox&layer=http://crossforest.eu/ilu/data/layer/original&soilUse=https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Use110&lngwest=0&lngeast=3&latnorth=40&latsouth=37&limit=5&offset=0`



10. Further information

This API has been built with CRAFTS (Configurable RESTful APIs For Triple Stores)

The CRAFTS API is documented here