

# 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:

https://crafts.gsic.uva.es/apis/globaleco/ Save ✎ 💬

**GET** ▼ https://crafts.gsic.uva.es/apis/globaleco/ Send ▼

**Params** Authorization Headers (5) Body Pre-request Script Tests Settings Cookies

Query Params

KEY	VALUE	DESCRIPTION	...	Bulk Edit
Key	Value	Description		

**Body** Cookies Headers (9) Test Results 🌐 401 Unauthorized 568 ms 396 B Save Response ▼

**Pretty** Raw Preview Visualize **JSON** ▼ ☰ 🔍

```
1 {
2   "status": 401,
3   "message": "Unauthorized: Authorization header required"
4 }
```

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 screenshot shows a REST client interface. At the top, the URL is `https://crafts.gsic.uva.es/apis/globaleco/`. The request method is `GET`. The Authorization tab is active, showing a Bearer token: `c41b6cd1-9ec5-40cf-9274-d3fd9151b67d`. Below the request configuration, the response body is displayed in JSON format:

```

1  {
2    "apiId": "globaleco",
3    "endpoints": [
4      {
5        "id": "crossforest",
6        "sparqlURI": "https://forestexplorer.gsic.uva.es/sparql/",
7        "graphURI": "http://crossforest.eu",
8        "httpMethod": "GET"
9      },
10     {
11       "id": "dbpedia",
12       "sparqlURI": "http://dbpedia.org/sparql",
13       "graphURI": "http://dbpedia.org",
14       "httpMethod": "GET"
15     }
16   ],
17   "model": [
18     {

```

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:

GET ▼ <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> Send ▼

Params ● Auth ● Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	id	Tree			
<input checked="" type="checkbox"/>	iri	https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tre...			
	Key	Value	Description		

Body Cookies Headers (1) Test Results 200 OK 93 ms 538 B Save Response ▼

Pretty Raw Preview Visualize JSON ▼ 🔍

```

1  {
2    "iri": "https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tree/05-0810-A-4-11",
3    "species": "https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43",
4    "dbh1mm": 224,
5    "dbh2mm": 228,
6    "heightM": 15.5,
7    "position": {
8      "iri": "https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/position/05-0810-A-4-11-23030-4326",
9      "latWGS84": 40.384187,
10     "lngWGS84": -4.697652
11   },
12   "plot": "https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/plot/05-0810-A-4"
13 }

```

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`

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

Params Authorization Headers Body Pre-request Script Tests Settings Cookies

Query Params

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	id	Species			
<input checked="" type="checkbox"/>	iri	https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43			

Body Cookies Headers (1) Test Results 200 OK 160 ms 5.14 KB Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "iri": "https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species43",
3    "scientificName": {
4      "la": "Quercus pyrenaica"
5    },
6    "vulgarName": [
7      {
8        "en": "Pyrenean oak"
9      },
10     {
11       "es": "Rebollo"
12     },
13     {
14       "de": "Pyrenäen-Eiche"
15     },
16     {
17       "fr": "Chêne tauzin"
18     },
19     {
20       "pt": "Quercus pyrenaica"
21     }
22   ],
23   "wikipediaPage": "https://en.wikipedia.org/wiki/Quercus_pyrenaica",
24   "superTaxon": "https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Genus121",
25   "dbpedia": {
26     "iri": "http://dbpedia.org/resource/Quercus_pyrenaica",
27     "comment": [
28       {
29         "ca": "El reboll o roure reboll (Quercus pyrenaica) és un roure present en molts llocs de la península Ibèrica però que, per singular capritx taxonòmic, duu aquest nom científic tot i no haver-n'hi al Pirineu. A Catalunya, aquesta espècie només es pot trobar al Bosc de Poblet. El nom comú, més encertat, ve de la capacitat de rebrotat quan se'l

```

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-](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/05-0810-A-4-18)

[18&iris=https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tree/06-0035-A-1-](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/06-0035-A-1-5)

[5&iris=https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tree/05-0093-A-3C-6](https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/ifn/tree/05-0093-A-3C-6)

The screenshot shows a REST client interface with a GET request to `https://crafts.gsic.uva.es/apis/globaleco/resources?id=Tree&iris=https://datos.iepnb.es/recurso/sector-pt...`. The response is a JSON array of two objects. The first object represents a tree resource with the following fields: `iri` (a URL to a specific tree), `species` (a URL to a species definition), `dbh1mm` (133), `dbh2mm` (145), `heightM` (10), and `position` (an object with `iri`, `latWGS84`, and `lngWGS84`). The second object is similar but for a different tree resource.

## 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>

The screenshot shows a REST client interface with a GET request to `https://crafts.gsic.uva.es/apis/globaleco/query?id=allSpecies`. The response status is 200 OK, with a response time of 150 ms and a size of 74.68 KB. The response body is displayed in JSON-LD format, showing a list of species results.

KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	id	allSpecies		

```

10  },
11  "results": {
12    "distinct": false,
13    "ordered": true,
14    "bindings": [
15      {
16        "species": {
17          "type": "uri",
18          "value": "https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Genus213"
19        },
20        "sciname": {
21          "type": "literal",
22          "xml:lang": "la",
23          "value": "Abies"
24        },
25        "esnames": {
26          "type": "literal",
27          "value": "Abetos"
28        },
29        "ennames": {
30          "type": "literal",
31          "value": "Firs"
32        }
33      },
34      {
35        "species": {
36          "type": "uri",
37          "value": "https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species31"
38        },
39        "sciname": {
40          "type": "literal",
41          "xml:lang": "la",
42          "value": "Abies alba"

```

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





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](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>

The screenshot shows a REST client interface with a GET request to <https://crafts.gsic.uva.es/apis/globaleco/query?id=numericTreeProps>. The response is a JSON array of two objects, each representing a tree property. The first object has an IRI for `hasDBH2InMillimeters` and a Spanish description. The second object has an IRI for `hasTotalHeightInMeters` and a Spanish description.

```
94 {
95   "propuri": {
96     "type": "uri",
97     "value": "https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/
           hasDBH2InMillimeters"
98   },
99   "esname": {
100     "type": "literal",
101     "xml:lang": "es",
102     "value": "tiene DN en milímetros"
103   },
104   "enname": {
105     "type": "literal",
106     "xml:lang": "en",
107     "value": "has DBH in millimeters"
108   }
109 },
110 {
111   "propuri": {
112     "type": "uri",
113     "value": "https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/
           hasTotalHeightInMeters"
114   },
115   "esname": {
116     "type": "literal",
117     "xml:lang": "es",
118     "value": "tiene altura total en metros"
119   },
120   "enname": {
121     "type": "literal",
122     "xml:lang": "en",
123     "value": "has total height in meters"
124   }
}
```

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/s<
  \"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

The screenshot shows a REST client interface. At the top, a GET request is defined with the URL: `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`. Below the URL bar, there are tabs for Params, Authorization, Headers, Body, Pre-request Script, Tests, Settings, and Cookies. The Params tab is active, showing two parameters: 'id' with value 'Patch' and 'iri' with value 'https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/mfe/patch/s5-0...'. The Body tab is also active, showing the response in JSON format. The response is a JSON object with the following structure:

```
1  {
2    "iri": "https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/mfe/patch/s5-02-1394795",
3    "soilUse": "https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Use111",
4    "canopyCoverTotalPercent": 60,
5    "canopyCoverTreesPercent": 35,
6    "polygon": {
7      "iri": "https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/mfe/polygon/
8        s5-02-1394795-4326",
9      "westBoundWGS84": -1.203881,
10     "eastBoundWGS84": -1.202393,
11     "northBoundWGS84": 38.74896,
12     "southBoundWGS84": 38.748091,
13     "areaM2": 5268.6192512,
14     "wkt": "POLYGON((-1.202492 38.748481,-1.202393 38.74896,-1.203235 38.748754,-1.203881 38.748091,-1.
15       202492 38.748481))",
16     "layer": "http://crossforest.eu/ilu/data/layer/s5"
17   },
18   "province": {
19     "iri": "https://datos.iepnb.es/recurso/sector-publico/territorio/Provincia/02",
20     "label": {
21       "es": "Albacete"
22     }
23   },
24   "infoSpecies": {
25     "iri": "https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/mfe/infoSpecies/24-3-90",
26     "percOccupation": 90,
27     "species": "https://datos.iepnb.es/def/sector-publico/medio-ambiente/ifn/Species24"
28   }
29 }
```

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`

The screenshot shows a REST client interface with a GET request and its response. The request is as follows:

```
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
```

The response is a JSON object:

```
14  "results": {
15    "distinct": false,
16    "ordered": true,
17    "bindings": [
18      {
19        "patch": {
20          "type": "uri",
21          "value": "https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/mfe/patch/original-06-1394834"
22        },
23        "poly": {
24          "type": "uri",
25          "value": "https://datos.iepnb.es/recurso/sector-publico/medio-ambiente/mfe/polygon/original-06-1394834-4326"
26        },
27        "west": {
28          "type": "typed-literal",
29          "datatype": "http://www.w3.org/2001/XMLSchema#decimal",
30          "value": "-5.411435"

```



## 10. Further information

---

This API has been built with [CRAFTS \(Configurable RESTful APIs For Triple Stores\)](#)

The CRAFTS API is documented [here](#)