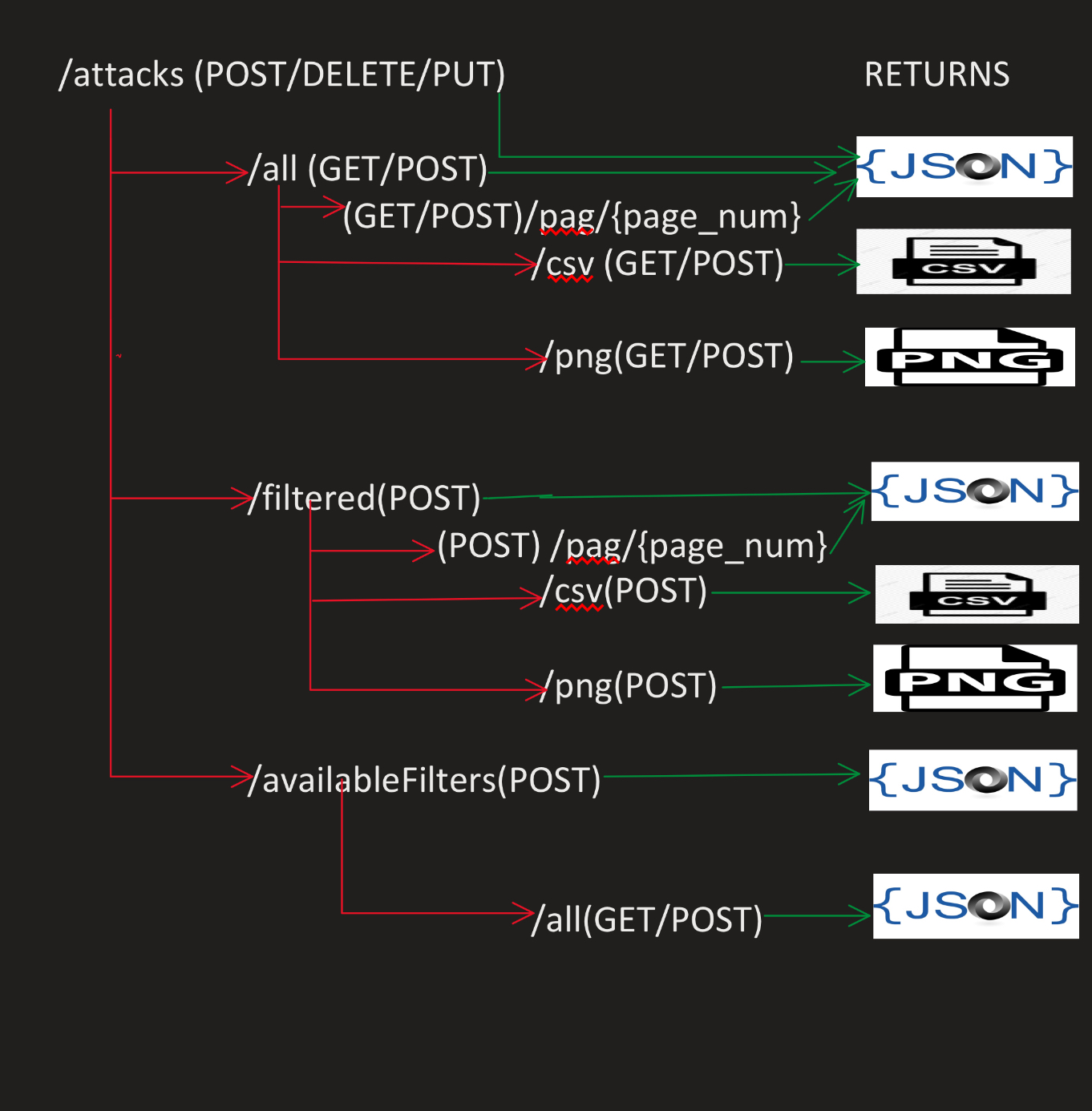
1.1 PROJECT STRUCTURE

The project contains 3 main parts:

* REST API, providing attacks data
* DATA VISUALIZER APP BACKEND
* DATA VISUALIZER APP FRONTEND

1.2 API STRUCTURE

ENDPOINTS & RESPONSE TYPES



1.2 API CONTROL FLOW

1 “/attacks”(POST):

Used for creating an attack. A post request shall be created,

where the content shall be a json describing the new attack and the admin\_token for manipulating data on server.

An attack can be described by a json where key = “attack\_property\_name”, value = “attack\_property\_value”. Properties describing an attack wich are not in the JSON, will be set to NULL.

This is a payload example for creating an attack which has eventid = 1:

{

"eventid":"1",

"admin\_token":"some\_key"

}

And a full query example with curl

curl -d '{"eventid":"1","admin\_token":"some\_key"}' -H "Content-Type: a pplication/json" -X POST <http://192.168.64.2/attacks>

For both success or failure of this query, the response will be a JSON

containing a status code and a description.

EG.

{"status": 200, "description":"object created"}

If the query failed due to bad token, the response will look like:

{"status": 400, "description": "Bad request, admin\_token not valid"}

2 “/attacks”(PUT)

Used for updating attacks, the JSON payload shall contain an array describing for which attacks to update a certain value(key = “for\_attacks\_matching”, value = json\_describing\_filters), “values\_to\_update” -> array of values to be updated, admin\_token -> the admin key, for updating server data.

Here is a query example, which updates iyear to 1999, for every attack which took place between 1970 - 2000 or 2001 - 2002 or in 2010 or in 2011 or in 2012 and it happend in Germany or Italy

curl -d '{

"for\_attacks\_matching":{

"intervals":{"iyear":[1970,2000,2001, 2002]},

"iyear":[2010,2011,2012],

"country\_txt":["germany", "italy"]

},

"admin\_token":"some\_key",

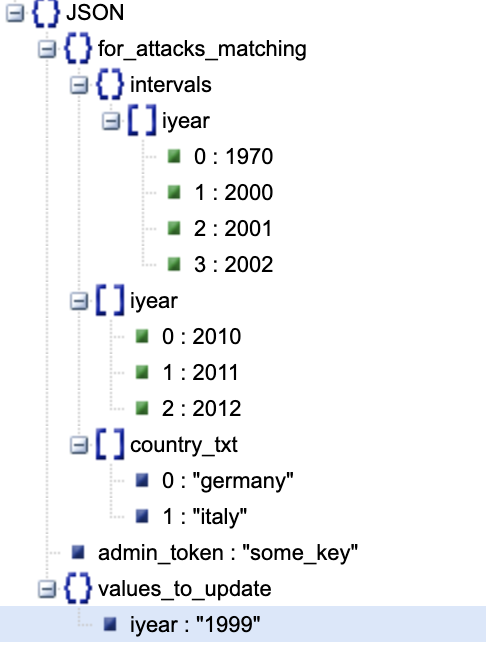
"values\_to\_update":{

"iyear":"1999"

}

}' -H "Content-Type: application/json" -X PUT http://192.168.64.2/attacks

At higher level, the JSON payload can be viewed like here:



The response is a JSON containing a response code and a description, Eg.

{"status:": 200, "description":"object updated"}

3 “/attacks”(DELETE)

Used for deleting attacks which are matching a filter criteria. Therefore, the JSON Payload shall also contain the “for\_attacks\_matching” field, and the admin\_token for updating server data.

Here is an example of a query for deleting attacks having eventid = 1.

curl -d '{

"for\_attacks\_matching":{

"eventid":[

"1"

]

},

"admin\_token":"some\_key"

}' -H "Content-Type: application/json" -X DELETE <http://192.168.64.2/attacks>

The output is the same as POST/PUT: status + description

4. “attacks/all” (POST/GET)

A simple GET/POST request to this endpoint will cause the API to respond with a JSON describing an array consisting of all attacks

Hence these doesn’t need any kind of filtering, POST request is also available, and the JSON payload won’t be taken into consideration.

Internally, these are stored as an associative array between an index and another asoociative array, EG:

attacks{

0 => array(130) {“country\_txt”:”germany”,”iyear”:1990, ect.}

1 => …

.

.

n => ...

“dataSz” => n - 1

}

Given this representation, the JSON will look the same, eg:





5. “attacks/all/pag/{page\_number}” (POST/GET)

Same as point 5, but it returns only one page of data, a page contains 3500 attacks, page number indicate which page you want.

6. “attacks/all/csv”(POST/GET)

Sends a csv file describing the attacks

7. “attacks/all/png”

Sends a png cartographic map representation of the attacks.

8. “attacks/filtered”(POST)

Given some user filters, returns the attacks matching the filters

The JSON payload contains the user filters, in the folowing form:

array() {  
 intervals:{“field1”:[interval\_left, interval\_right, interval\_left\_1, interval\_right\_1, ..]},

“field\_name”:”field\_value”

.

.

.

}

Payload json example for selecting all atacks between 1970 and 200, or 2001 and 2002 or in 2010,oe 2011, or 2012

{

"intervals": {

"iyear": [

1970,

2000,

2001,

2002

]

},

"iyear": [

2010,

2011,

2012

],

"country\_txt": [

"germany",

"italy"

]

}

An example of a query for all atacks between 1970 - 200 or 2001-2002, or 2010,or 2011, or 2012, and happened in germany or italy.

curl -d '{"intervals":{"iyear":[1970,2000,2001, 2002]}, "iyear":[2010,2011,2012], "country\_txt":["germany", "italy"]}' -H "Content-Type: application/json" -X POST http://192.168.64.2/attacks/all/pag/1

the response is the same as point 4.

9. “attacks/filtered/pag/{page\_number}”(POST)

Same as point 8, but it returns only one page of data, a page contains 3500 attacks, page number indicate which page you want.

9. “attacks/filtered/csv”(POST)

Returns the filtered attacks as csv representation.

9. “attacks/filtered/png”(POST)

Returns the filtered attacks as cartographic representation.

10.”attacks/availablefilters”(POST)

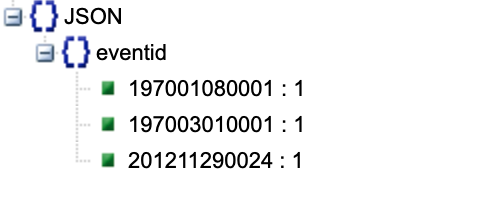
For a given set of filters, it returns all possible values for each filter.

The JSON Payload shall be formatted the same as for point 8.

The response is structured internally as an associative array:

array() => {“field\_name” => array()=>{“field\_value”:”number \_of\_occurences”}}, this structure implements the behaviour of a set.

More explicit, here is a possible json response only for the eventid field:



{

"eventid": {

"197001080001": 1,

"197003010001": 1,

"201211290024": 1

}

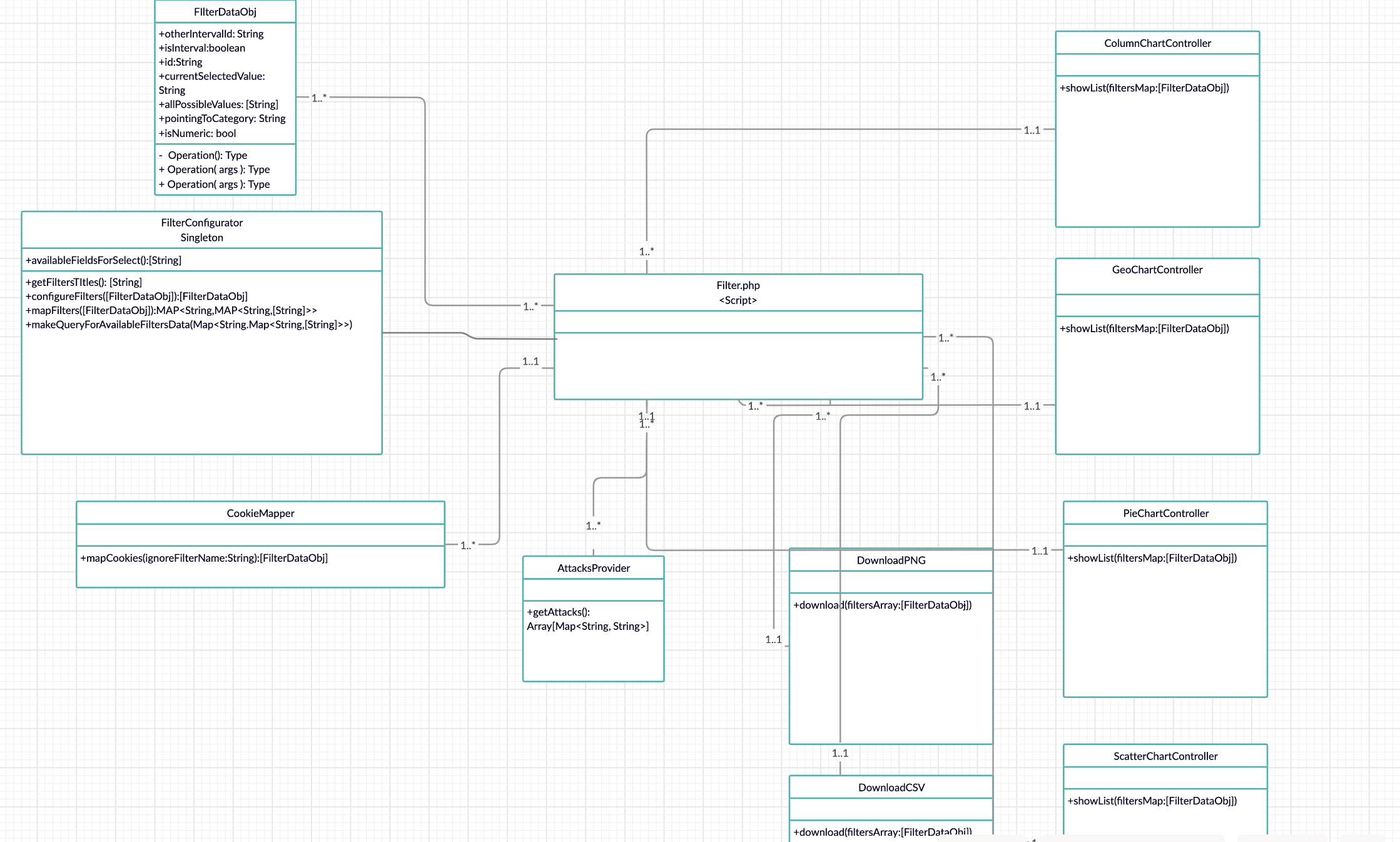
}

11.”attacks/availablefilters”(POST/GET)

Same behaviour as point 10, but it always returns all filters values, it doesn’t need filters input, hence for a post request, the payload won’t be taken into consideration.

2.1 APP BACKEND

The app backend design is implemented as follows from the UML diagram.



Every filters added by an user are persisted into cookies.

2.2 APP FRONTED

Divided in to separate views, templated with twig engine.