The EHRI GraphQL API

IEEE Big Data Workshop on Computational Archival Science

13/12/2017
Mike Bryant

CONNECTING COLLECTIONS
The EHRI Project

The main objective of EHRI is to support the Holocaust research community by

1. **integrating** information on key archival collections and institutions into an online portal ([https://portal.ehri-project.eu](https://portal.ehri-project.eu))

2. **encouraging** collaborative Holocaust research and archiving and investigating new methodologies

[http://ehri-project.eu](http://ehri-project.eu)
EHRI Portal APIs – why?

APIs = ways of getting data in structured form

We want to offer our data to researchers in structured form

We want to facilitate data interchange data between our own sites

Also: Experimentation!
During my fellowship at The Wiener Library, I benefited from the sources of United Nations War Crimes Commission (UNWCC, here i.e. Committee I: Facts and Evidences, Committee III: Legal Committee, Correspondence with National Office Poland, Polish Trial Reports) regarding Polish activities within the UNWCC.

The second category of sources I could also work on was the impressive collection of Raphael Lemkin papers. While the originals are held at the New York Public Library, The Wiener Library holds a microfilm version of the collection, which largely document Lemkin’s intense interest in the subject of genocide. I was particularly interested in the correspondence with public officials, newspapers, academics, and
Search API – REST-style: http://jsonapi.org

Global Search

The Global search action (/search) allows you to search all available item types. Four parameters are currently supported:

- **q**: A text query, following the same rules as searching on the portal site.
- **type**: One of the available data types. Can be used multiple times.
- **page**: Since results are paginated, this number selects the desired page.
- **limit**: The number of results to fetch per page, up to a maximum of 100.

Test it!

<table>
<thead>
<tr>
<th>Query</th>
<th>Type</th>
<th>Page</th>
<th>Per page</th>
<th>Click to test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All</td>
<td>1</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

Global Search Example Results

```
{
  "data": [
    {
      "id": "at-001903-110003504",
      "type": "DocumentaryUnit",
      "attributes": {
        "localId": "110003504",
        "alternateIds": [],
        "descriptions": [
          {
            "localId": "deu",
            "languageCode": "deu"
          }
        ]
    }
  ]
}
```
This XML file does not appear to have any style information associated with it. The document tree is shown below.

```xml
  <responseDate>2017-12-08T03:23:01Z</responseDate>
  <request verb="Identify">https://portal.ehri-project.eu/api/oaipmh</request>
  <Identify>
    <repositoryName>
      European Holocaust Research Infrastructure Portal Repository
    </repositoryName>
    <baseURL>https://portal.ehri-project.eu/api/oaipmh</baseURL>
    <protocolVersion>2.0</protocolVersion>
    <adminEmail>info@ehri-project.eu</adminEmail>
    <earliestDatestamp>2013-09-09T1:31:51Z</earliestDatestamp>
    <deletedRecord>persistent</deletedRecord>
    <granularity>YYYY-MM-DDThh:mm:ssZ</granularity>
    <compression>gzip</compression>
  </Identify>
</OAIPMH>
```
XML Downloads (EAD / EAC / EAG)

Joodsche Raad voor Amsterdam

Identifier: MF759708
Language of Description: Dutch
Dates: 1940-1945
Level of Description: Fonds
Source: EHRI Partner

Extent and Medium: 7,9 meter (305 inventarisnummers)

Biographical History:

Geschiedenis

Op 12 februari 1941 gaf de Beauftragter des Reichskommissars für die Stadt Amsterdam, Böhmker, aan Abraham Asscher (diamant-industriëel te Amsterdam) opdracht onmiddellijk een "Judenaat" te vormen, die uit twintig personen zou moeten bestaan. Asscher was van mening, dat bij het samenstellen van die raad tevens prof.dr. David Cohen diende te worden betrokken, die hij sinds jaren uit tal van besturen en commissies kende; deze had voorts o.m. als voorzitter van het Joodsche Vluchtelingencomité van 1933 af een intensief contact gehad met Nederlandse autoriteiten. Cohen verklaarde zich bereid aan de totstandkoming van de raad mee te werken en beiden deden op 12 en 13 februari daartoe een aantal uitnodigingen uitgaan, welke door zeventien personen werd aanvaard.

De Joodsche Raad: met de armen over elkaar aan tafel zit Asscher, aan zijn linker zijde zit Cohen.
Ad-hoc datasets

EHRI Datasets

These datasets are dynamically-generated structured data about EHRI’s collections available in JSON, CSV, and TSV formats.

Concepts and Historical Agents
A unified CSV of concept names and historical agents
json / csv / tsv

Repository Info
Names, URLs, country, and email addresses for repositories.
json / csv / tsv

EHRI SKOS terms / labels
### Stutthof concentration camp (Q326193)

German Nazi concentration camp outside of Danzig (Gdańsk) in present-day Poland

**Konzentrationslager Stutthof / KZ Stutthof**

#### In more languages

<table>
<thead>
<tr>
<th>Language</th>
<th>Label</th>
<th>Description</th>
<th>Also known as</th>
</tr>
</thead>
<tbody>
<tr>
<td>English</td>
<td>Stutthof concentration camp</td>
<td>German Nazi concentration camp outside of Danzig (Gdańsk) in present-day Poland</td>
<td>Konzentrationslager Stutthof, KZ Stutthof</td>
</tr>
<tr>
<td>French</td>
<td>camp du Stutthof</td>
<td>No description defined</td>
<td>Camp de Stutthof, Stutthof, Stutt, Kl. Stutthof</td>
</tr>
<tr>
<td>German</td>
<td>KZ Stutthof</td>
<td>Konzentrationslager</td>
<td>Konzentrationslager Stutthof</td>
</tr>
<tr>
<td>Scots</td>
<td>No label defined</td>
<td>No description defined</td>
<td></td>
</tr>
</tbody>
</table>

#### Statements
Why another API?

Lots of EHRI data not exposed by previous interfaces:
• Interconnections / links between items
• Annotations
• Digital provenance

REST-like Search API cumbersome to extend and maintain

Needed something better suited to building alternative Uis

Also: something that was more user-friendly…?
GraphQL?

• Invented by Facebook in 2012
• **Not** related to *graph databases!*
  • Name refers to a generic object graph
  • Not tied to any type of storage mechanism
• **Schema based**
  • Uses a schema to define the data types that are available and validate requests
  • Allows requests to precisely defining the data required in the response
  • Introspection: standard requests can inspect the schema
• Somewhere between HTTP REST API and full DB query language
GraphQL query & response example

```graphql
query example1 {
  repository(id: "us-005578") {
    name
    address {
      city
    }
    itemCount
  }
}
```

```json
{
  "data" {
    "repository" {
      "name": "USHMM",
      "address": {
        "city": "Washington_D.C."
      }
    }
    "itemCount": 9574
  }
}
```

Queries list data fields required from those defined in the schema.

Optional or mandatory parameters can qualify certain fields.

Output data corresponds one-to-one with query.
GraphQL – ecosystem

Big ecosystem of libraries and tools, supported by companies such as Facebook and Github:

- Apollo
- Relay
- Graphcool
- graphql-java
- … and many more

- GraphiQL
  - Online query-editing interface
  - Works with any “standard” GraphQL API end-point
GraphiQL – Inline documentation...

```graphql
query example($id: ID!) {
  DocumentaryUnit(id: $id) {
    description {
      scopeAndContent
      biographicalHistory
    }
  }
}
```

**archivistNote**: String

To explain how the description was prepared and by whom.

**archivalHistory**: String

Provides information on the history of the unit of description that is significant for its authenticity, integrity and interpretation.

**acquisition**: String

To identify the immediate source of acquisition or transfer.

**appraisal**: String

To provide information on any appraisal, destruction and scheduling action.

**accruals**: String

To inform the user of foreseen additions to the unit of description.

**biographicalHistory**: String

Provides an administrative history of, or biographical details on, the creator (or creators) of the unit of...
GraphiQL – Auto-completion of fields...
GraphQL – Inline error messages...

query example($id: String!)
{
  DocumentaryUnit(id: $id)
  {
    description {
      scopeAndContent
      biographicalHistory
    }
  }
}

Unknown argument "foo" on field "DocumentaryUnit" of type "Root".

The description's title

TYPE
String
The EHRI GraphQL API - implementation

Runs as a plugin to EHRI’s Neo4j-based database

- Minimal network overhead
- Index-free traversals and native property lookups

Inherits access-control system used in the EHRI portal

GraphQL schema quite similar to database schema, but de-coupled and simplified in various ways.
Use Case – retrieving annotations

Acquisition
Donated by Vivien Korn

Scope and Content
This collection consists of the papers of the Toch and Korn families, Jewish refugees from Vienna. Whilst the children Erika and Harry Toch emigrated to England and Palestine, respectively to flee Nazi persecution, their parents Wilhelm and Margarethe Toch were deported to Terezin concentration camp from where only their mother returned. Erika got married to Polish refugee, Salman Korn, in 1941 whose papers and correspondence are also included. The collection also contains the papers of Wilhelm Toch’s brother, Moritz Toch and his wife Else, who also emigrated to the UK in 1939.

Second line of this text should be “She had come over earlier in 1939”
- Howard Falksohn 3 months ago Demote

NB: This text, as of 5th Sept 2017, contains two misspellings of “Toch” as “Koch”.
- Mike Bryant 3 months ago Demote
Use Case – retrieving annotations

```
query getAnnotations($id: ID!) {
  DocumentaryUnit(id: $id) {
    annotations {
      id
      body
      field
      by
    }
  }
}
```

```
{
  "data": {
    "DocumentaryUnit": {
      "annotations": [
        {
          "id": "6ad96618-921c-11e7-94f8-d7ab846fe93f",
          "body": "Second line of this text should be "She had come over earlier in 1939"",
          "field": "biographicalHistory",
          "by": "Howard Falksohn"
        },
        {
          "id": "f66325be-9219-11e7-94f8-d7ab846fe93f",
          "body": "NB: This text, as of 5th Sept 2017, contains two misspellings of "Toch" as "Koch.",
          "field": "biographicalHistory",
          "by": "Mike Bryant"
        }
      ]
    }
  }
}
```
Archival “resources” are hierarchical, not a flat set

- Parent item
- Child items

This can cause problems of over-fetching or under-fetching of data in resource-oriented (REST-like) APIs:

How much context is integral to the resource?

vs

How much is needed for any given usage?
Use Case - context

```graphql
query example($id: ID!) {
  DocumentaryUnit(id: $id) {
    repository {
      id
      description(languageCode: "eng") {
        name
      }
    }
    ancestors {
      id
      description(languageCode: "eng") {
        name
      }
    }
    description {
      name
      scopeAndContent
    }
  }
}
```

```
{ "data": {
  "DocumentaryUnit": {
    "repository": {
      "id": "dk-002313",
      "description": {
        "name": "Rigsarkivet"
      }
    },
    "ancestors": [
      {
        "id": "dk-002313-københavns_ams_nordre_birks_politi",
        "description": {
          "name": "Copenhagen County North Birks (Court) Police"
        }
      },
      {
        "id": "dk-002313-københavns_ams_nordre_birks_politi",
        "description": {
          "name": "File list, political proceedings 1945-1950, pcl. XII.c-1
        }
      }
    ],
    "description": {
      "name": "File list, political proceedings",
      "scopeAndContent": "File list, political proceedings 1945-1950, pcl. XII.c-1"
    }
  }
}
```
Use Case – Bulk data streaming

Most APIs are batch-oriented / paginated

Fetching bulk data means fetching successive pages in multiple requests

Many Big Data systems are better suited to stream processing rather than in-memory or distributed batches
Use Case – Bulk data streaming

EHRI GraphQL API allows setting a “streaming” mode

Disables default pagination, allowing a single request to stream potentially unlimited amounts of data

Very useful when combined with a JSON stream processing tool such as JQ

curl --header X-Stream:true --data-binary @query.gql \https://portal.ehri-project.eu/api/graphql
**EHRI GraphQL: limitations**

Not nearly as expressive as a bona-fida database query language:

- Cannot do aggregation, sorting, or search
  - Flip-side to allowing streaming bulk requests, available operations are deliberately limited in complexity to avoid potential resource-usage problems

Currently only supports JSON response format
Future work...

Add lots more examples that users can load in GraphiQL.

Allow users to download data as a file, from GraphiQL.

Improve schema in certain ways...
In summary

No API is perfectly suited to every task

GraphQL has many advantages

• “Bring your own storage”: can integrate with legacy databases
• Query model handles archival hierarchy/context well
• Schema and introspection enable many user aids and make API more “explorable” and discoverable
• Very active ecosystem of libraries and tools
THANKS!