

Figshare API Workshop

Open Repositories 2023
Stellenbosch, South Africa
Twitter: #openrepos2023



Quick Activity



Photo by [iorni.com](#) on [Unsplash](#)



showcase your institution's research outputs in one place

Instructors



**Adrian-Tudor
Pănescu**
Technical Team
Leader



**Andrew
Mckenna-Foster**
Product Specialist
(Assistant instructor)



Photo by [Neil Thomas](#) on [Unsplash](#)

This workshop is an opportunity to create something useful to you

Examples

A personal statistics report

Try this in [Google Colab](#).

This retrieves all metadata and statistics for an author - items and collections

There are two ways to collect records: by name or by ORCID

Import libraries

```
In [1]:  
import json  
import requests  
import pandas as pd  
import csv  
import datetime
```

Set base URL

```
In [2]:  
#Set the base URL  
BASE_URL = 'https://api.figshare.com/v2'
```

Retrieve Metadata by Author Name (Note this does not disambiguate people with the same name)

```
In [4]:  
#author name  
name = "ENTER NAME BETWEEN QUOTES"
```

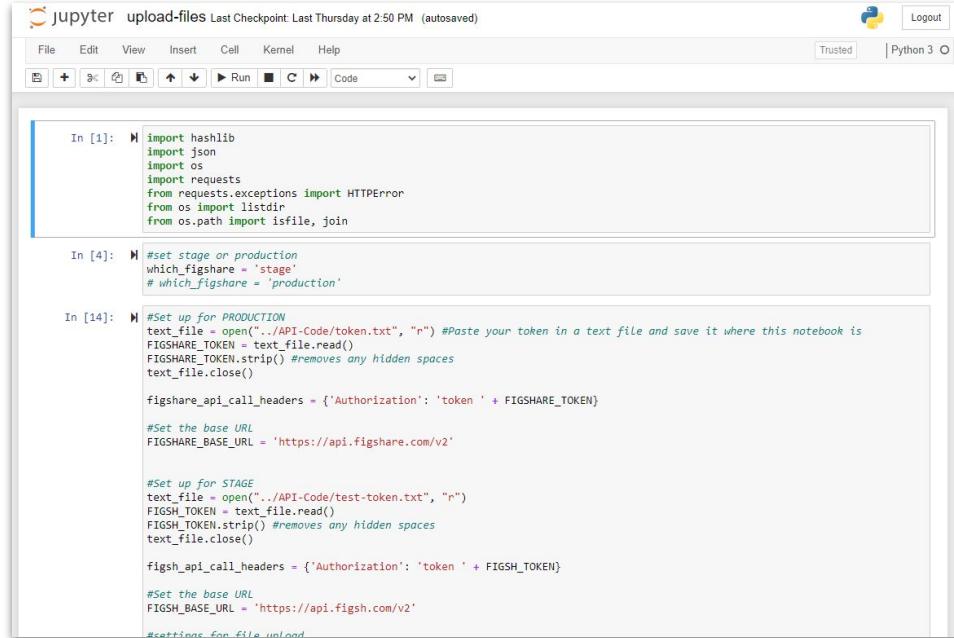
```
In [5]:  
#Retrieve List of metadata  
#SET THE PAGE SIZE to make sure you get all the records  
  
#Gather basic metadata for items (articles) that meet your search criteria  
query = "https://api.figshare.com/v2/items?size=1000" #Set up string
```



Examples

Manage repository records

Wesleyan University uses the API to manage records and upload large file collections



The screenshot shows a Jupyter Notebook interface with the title "jupyter upload-files Last Checkpoint: Last Thursday at 2:50 PM (autosaved)". The notebook has three cells:

- In [1]:

```
import hashlib
import json
import os
import requests
from requests.exceptions import HTTPError
from os import listdir
from os.path import isfile, join
```
- In [4]:

```
#set stage or production
which_figsshare = 'stage'
# which_figsshare = 'production'
```
- In [14]:

```
#Set up for PRODUCTION
text_file = open("../API-Code/token.txt", "r") #Paste your token in a text file and save it where this notebook is
FIGSHARE_TOKEN = text_file.read()
FIGSHARE_TOKEN.strip() #removes any hidden spaces
text_file.close()

figshare_api_call_headers = {'Authorization': 'token ' + FIGSHARE_TOKEN}

#Set the base URL
FIGSHARE_BASE_URL = 'https://api.figshare.com/v2'

#Set up for STAGE
text_file = open("../API-Code/test-token.txt", "r")
FIGSH_TOKEN = text_file.read()
FIGSH_TOKEN.strip() #removes any hidden spaces
text_file.close()

figsh_api_call_headers = {'Authorization': 'token ' + FIGSH_TOKEN}

#Set the base URL
FIGSH_BASE_URL = 'https://api.figsh.com/v2'

#settings for file upload
```



Examples

Harvest records to create a data catalog

Either from another repository to Figshare or from Figshare to another repository

Macquarie University built a tool to harvest metadata from Dryad and deposit in their repository as linked file records

Dryad to Figshare harvest

Python 3 script to harvest metadata from a selected organisation's Dryad data records and generate matching metadata-only records in an institutional Figshare repository.

Notes

- This script only harvests metadata from dryad-based datasets from an institution.
- The existing (dryad-generated) dataset is harvested and converted into a generated Figshare record.
- The 'link file' option is used on the Dryad dataset to point to the Figshare record.
- This is 'one-time only' script - to use it again you will need to update the script to be an ongoing harvest.

The screenshot shows the Macquarie University Figshare interface. At the top, there is a navigation bar with 'Browse', a search bar 'Search on Macquarie University...', and 'Log in'. Below the navigation bar, there is a header for a harvested dataset: 'File(s) stored somewhere else' with a DOI link: <https://doi.org/10.5061/dryad.ct121>. A note below the header states: 'Please note: Linked content is NOT stored on Macquarie University and we can't guarantee its availability, quality, security or accept any liability.' The main content area displays the harvested dataset details: 'Data from: Developmental stress increases reproductive success in male zebra finches'. It includes a 'Cite' button, a 'Share' button, and a '+ Collect' button. Below these buttons, it says 'Dataset posted on 2022-06-10, 21:12 authored by Ondi L. Crino, Colin T. Prather, Stephanie C. Driscoll, Jeffrey M. Good, Creagh W. Breuner'. To the right, there is a 'USAGE METRICS' section showing '19 views', '0 downloads', and '0 citations'. A note at the bottom of the dataset page says: 'There is increasing evidence that exposure to stress during development can have...'

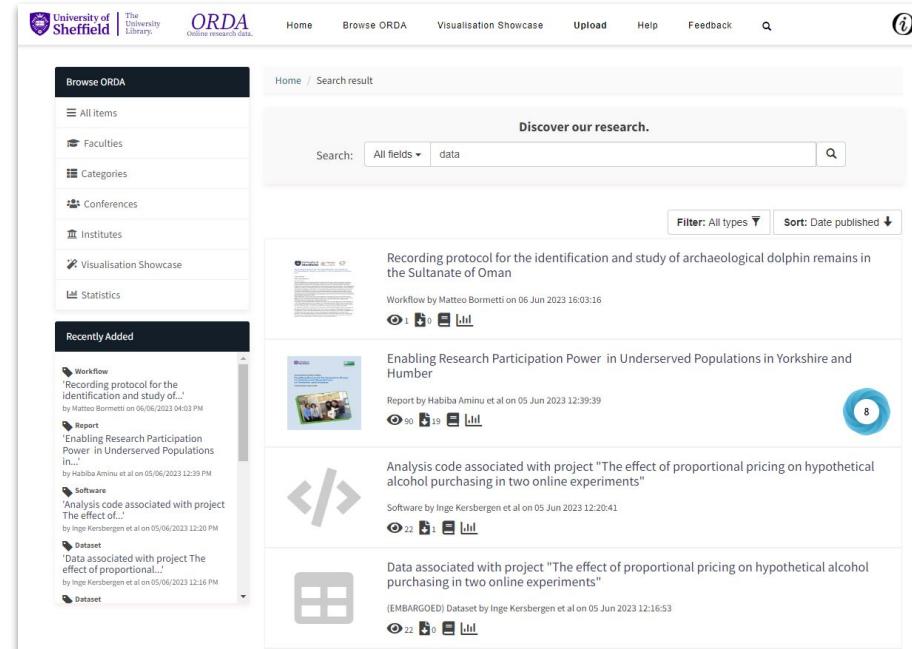
https://github.com/mq-eresearch/dryad_to_figshare/tree/v1.0.0



Examples

Build a custom web application

University of Sheffield built a custom search interface:
<https://orda.shef.ac.uk/>



The screenshot shows a custom web application interface for research data. At the top, there is a header with the University of Sheffield logo, the text 'The University Library.', and the 'ORDA' logo with the subtitle 'Online research data.' Below the header, a navigation bar includes links for 'Home', 'Browse ORDA', 'Visualisation Showcase', 'Upload', 'Help', 'Feedback', and a search icon. On the left, a sidebar titled 'Browse ORDA' contains links for 'All items', 'Faculties', 'Categories', 'Conferences', 'Institutes', 'Visualisation Showcase', and 'Statistics'. The main content area has a title 'Discover our research.' and a search bar with dropdowns for 'All fields' and 'data', and a search button. Below the search bar, there are filters for 'All types' and 'Sort: Date published'. The main content displays a list of research outputs:

- Recording protocol for the identification and study of archaeological dolphin remains in the Sultanate of Oman**
Workflow by Matteo Bornetti on 06 Jun 2023 16:03:16
View details
- Enabling Research Participation Power in Underserved Populations in Yorkshire and Humber**
Report by Habiba Aminu et al on 05 Jun 2023 12:39:39
View details
- Analysis code associated with project "The effect of proportional pricing on hypothetical alcohol purchasing in two online experiments"**
Software by Inge Kersbergen et al on 05 Jun 2023 12:20:41
View details
- Data associated with project "The effect of proportional pricing on hypothetical alcohol purchasing in two online experiments"**
(EMBARGOED) Dataset by Inge Kersbergen et al on 05 Jun 2023 12:16:53
View details



showcase your institution's research outputs in one place

figshare.com

@figshare

Please note down 1-2 projects you'd like to work on

As we progress through the workshop apply what you learn to your project

You may not finish the project today, but should leave with a clear roadmap



Photo by [No Revisions](#) on [Unsplash](#)



showcase your institution's research outputs in one place

[figshare.com](#)

@figshare

Workshop Website

<https://amckennafoster.github.io/figshare-api-workshop.github.io/index.html>

- Schedule
- Links
- Resources



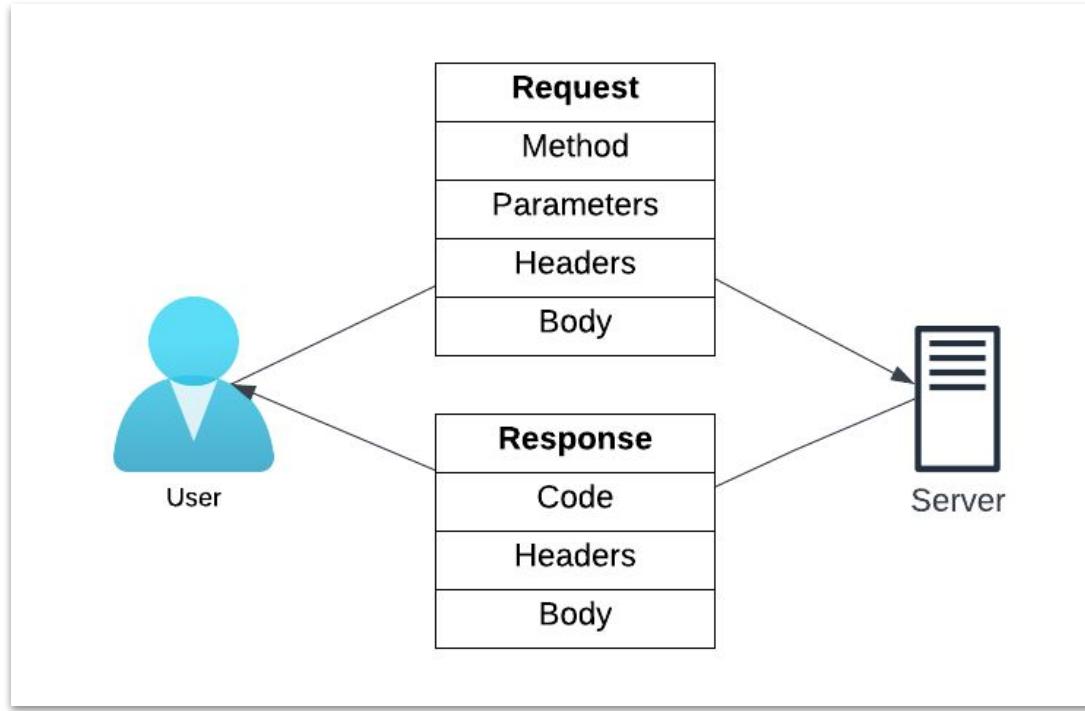
Workshop Outline

- Introduction to HTTP requests and APIs
- Introduction to Figshare's API(s)
- Accessing Figshare's REST API using Postman
- Building an application over Figshare's REST API



HTTP Requests and APIs

What is a HTTP request?



What is a HTTP request?

```
POST /v2/articles/search?offset=10&limit=1 HTTP/1.1
```

```
Host: api.figshare.com
```

```
Content-Type: application/json
```

```
Authorization: Bearer b2be49036a3158c5edd5a0553ae9
```

```
{"search_for": ":tags: test"}
```



HTTP is stateless

- Requests are independent from one another
- Each request needs to contain enough information in order for the server to be able to respond to it
- This is why, for example, you need to send authentication information with each HTTP request

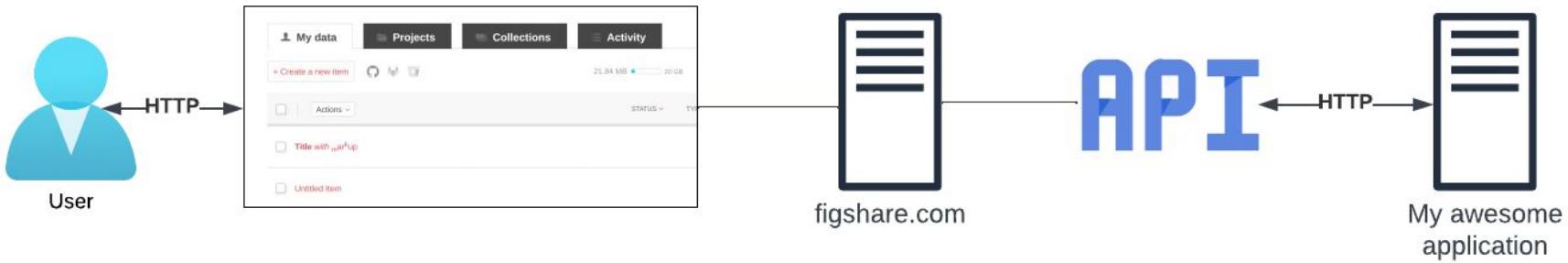


What is an API?

- Abbreviation for **Application Programming Interface**
- It's an interface which allows other applications to use the functionality of a software system



What is an API?



What is REST?

- Abbreviation for **REpresentational State Transfer**
- A set of principles on how web APIs should be built
- Servers should always respond with the representation of a *resource*



Example of a REST resource

```
{  
  "files": [  
    {  
      "id": 25133441,  
      "name": "Panescu_Advances_in_Digital_Repositories_PhD_Thesis.pdf",  
      "size": 1411929,  
      "is_link_only": false,  
      "download_url": "https://ndownloader.figshare.com/files/25133441",  
      "supplied_md5": "87f39b4dal371572523b59bf83bb32e",  
      "computed_md5": "87f39b4dal371572523b59bf83bb32e"  
    }  
,  
  "custom_fields": [],  
  "authors": [  
    {  
      "id": 1402906,  
      "full_name": "Adrian-Tudor Panescu",  
      "is_active": true,  
      "url_name": "Adrian-Tudor_Panescu",  
      "orcid_id": "0000-0002-8940-898X"  
    }  
,  
  "figshare_url": "https://figshare.com/articles/thesis/Advances_in_Digital.Repositories/8247626",  
  "description": "This thesis presents the evolution of research repositories, from collections of peer-reviewed outputs, such as journal articles or monographs, to collections holding a wide array of materials, including data sets, preprints, scientific software or protocols, based on the new realities of the scientific research endeavour. It also discusses a number of novel practical solutions to repository issues such as licencing, bibliographic record modelling, dissemination and linking, or record management and migration, by employing new technologies such as linked data, blockchain, and extract, transform and load frameworks.",  
  "funding": null,  
  "funding_list": [],  
  "version": 2,  
  "status": "public",  
}
```



What is JSON?

- Abbreviation for **JavaScript Object Notation**
- Human-readable data interchange format
- Consists of key-value pairs, lists and other basic types (numbers, strings)
- For example: { "name": "Jane Doe", "someList": [1, "two", 3] }



REST and request methods (verbs)

- GET: retrieve a resource
- POST: add a new resource
- PUT: update a new resource
- PATCH: partially update a new resource
- DELETE: remove a resource



HTTP status codes

- Standard (and easiest) way of understanding what happened with your requests
- Some common status codes:
 - 200 OK
 - 400 Bad request
 - 403 Forbidden
 - 404 Not Found
 - 500 Internal Server Error
- Full list: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Status>



HTTP request body

- The actual data sent to the server
- It's any sequence of bytes (JSON, XML, an image file) but...
- We need to tell the server what we are sending:
 - Content-Type: application/json
- The server response can also contain any sequence of bytes but...
- We can tell the server what we accept:
 - Accept: */*
- The request/response bodies are optional



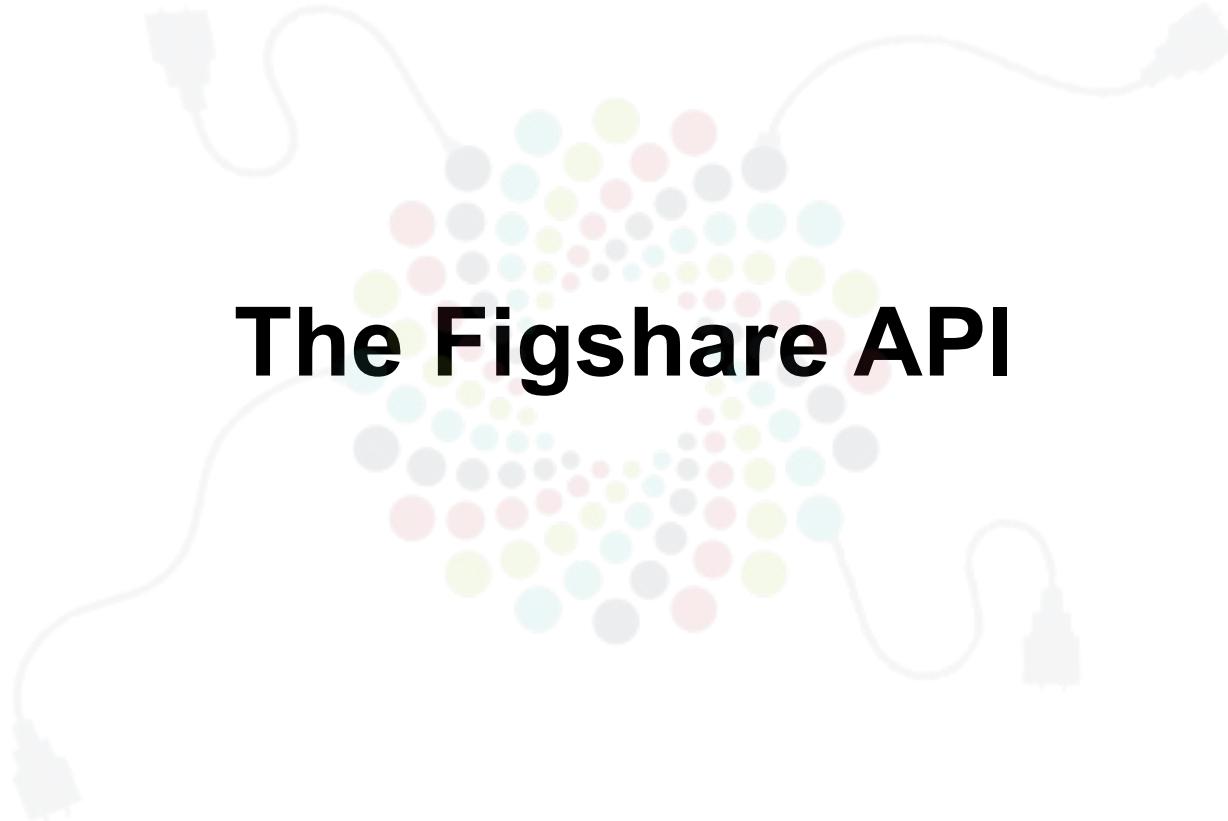
HTTPS

- Secure HTTP
- A secure channel is created between the client and the server
- Request details (URL, headers, body) are not visible on the open network
- Important if sensitive data (e.g., authentication credentials) is exchanged



Tea Break
30 min





The Figshare API

Figshare API(s)

- Figshare actually has 4 different *APIs*:
 - REST API: <https://api.figshare.com/v2>
 - Stats API: <https://stats.figshare.com/>
 - OAI-PMH: <https://api.figshare.com/v2/oai>
 - ResourceSync: <https://scholardata.sun.ac.za/.well-known/resourcesync>



Figshare API(s)

- REST API:
 - v1: not very RESTful, deprecated in 2021
 - v2: current version, centered around resources, backwards-compatible
 - v3: will better expose all UI functionality, improve documentation, better integrate stats
- Stats API:
 - Distinct due to the fact that stats are held in a different database and processed by different services than metadata
- OAI-PMH:
 - implemented in order to allow harvesting all Figshare items
- ResourceSync:
 - Another standard from OAI allowing synchronization between systems
 - Used for implementing sitemaps



ACTIVITY

Try GET request

Instructions to retrieve full metadata and retrieve views are here:

<https://amckennafoster.github.io/figshare-api-workshop.github.io/workshop/workshop-api-basics.html>

OpenAPI and Swagger

- OpenAPI
 - A specification language for REST APIs
 - Allows prototyping and generating documentation, client code and test cases
- Swagger:
 - OpenAPI implementation
 - Editor: <https://editor.swagger.io/>
 - Swagger UI: <https://swagger.io/tools/swagger-ui/>



Figshare and OpenAPI

- Documentation publicly available at <https://docs.figshare.com>
- Source code for documentation:
https://github.com/figshare/user_documentation
- Swagger file:
https://github.com/figshare/user_documentation/blob/master/swagger_documentation/swagger.json



API authentication

- REST API - personal token:
<https://help.figshare.com/article/how-to-get-a-personal-token>
 - OAuth2 - create applications which require users to log into their own Figshare account
- Stats API - user/password, contact support@figshare.com for an account
- Credentials are sent to the API server via the Authorization header
- **Make sure credentials are stored in a secure location!**



ACTIVITY

Authenticate and try a POST request

<https://amckennafoster.github.io/figshare-api-workshop.github.io/workshop/workshop-api-basics.html>

Sandbox accounts

global.user@figsh.com.bk

Bongani Jwara

Charl Roberts

Nambitha Manqola

Nkululeko Magwaza

global.user2@figsh.com.bk

Andisiwe Magocoba

Bubele Bido

Janina Van der Westhuizen

Yan Han

global.user3@figsh.com.bk

Samuel Simango

Songezo Mpikashe

Sizwe Ngcobo

global.user4@figsh.com.bk

Regina Sikhosana

Richard Nobebe

Xabiso Xesi

global.user5@figsh.com.bk

Eddie Mathiba

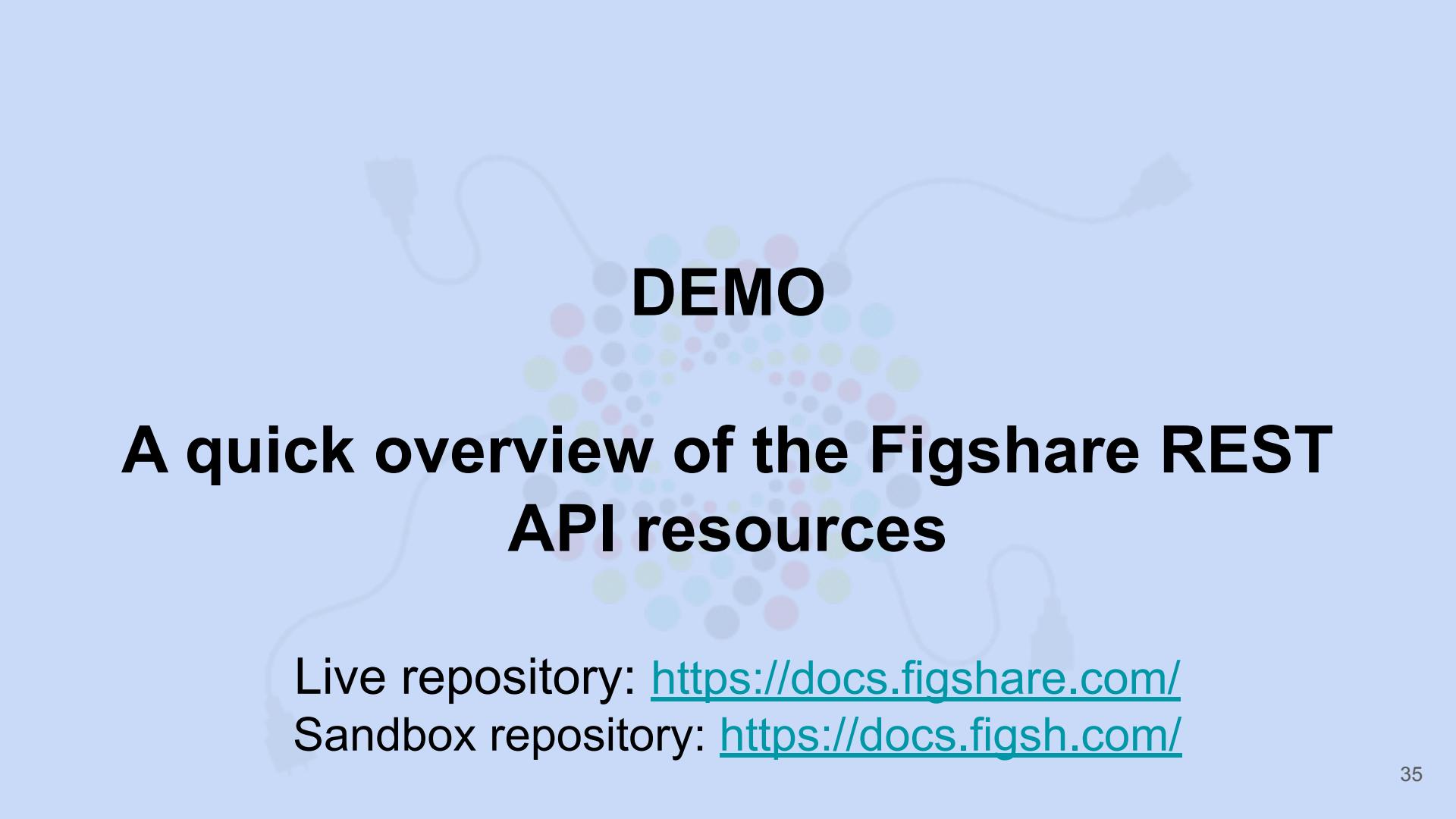
Martin Dreyer

Motlanalo Hlophe

Tshinakaho Malesa

<https://global.figsh.com>





DEMO

A quick overview of the Figshare REST API resources

Live repository: <https://docs.figshare.com/>

Sandbox repository: <https://docs.figsh.com/>

Q&A & PROJECT TIME

Use this time to ask questions and lay out the requirements for your project(s):

1. What is your objective
2. What information will you need to retrieve or send
3. What endpoints will you need to use



Lunch 1hr

Performing Figshare API requests

Postman

- Tool for performing API requests
- Can be downloaded from <https://www.postman.com/downloads/>
 - Web version available at <https://web.postman.co/>
- All demo requests available as a Postman collection:
<https://www.postman.com/tudor/workspace/public-workspace/collection/329625-1a755e1d-811a-48b8-8471-860e3f09f0ec>





DEMO

Using Postman to perform requests

<https://amckennafoster.github.io/figshare-api-workshop.github.io/workshop/postman-use-api.html>

Retrieve item metadata

The screenshot shows the Postman application interface. At the top, it displays the URL `https://api.figsh.com/v2/articles/8418909`. Below the URL, there are tabs for Params, Authorization, Headers (5), Body, Pre-request Script, Tests, Settings, and Cookies. The Headers tab is currently selected. Under the Params tab, there is a table titled "Query Params" with columns for Key, Value, Description, and Bulk Edit. The table has one row with the key "Key" and value "Value". In the main body area, there are tabs for Body, Cookies, Headers (12), and Test Results. The Body tab is selected, showing a JSON response. The response is displayed in a "Pretty" format with line numbers 1 through 17. The JSON structure includes fields for files, custom fields, and authors.

```
1 {
2   "files": [
3     {
4       "id": 830411360,
5       "name": "aes.JPG",
6       "size": 172621,
7       "is_link_only": false,
8       "download_url": "https://ndownloader.figsh.com/files/830411360",
9       "supplied_md5": "9a5d445f37c8c174007cd431f376101a",
10      "computed_md5": "9a5d445f37c8c174007cd431f376101a"
11    }
12  ],
13  "custom_fields": [],
14  "authors": [
15    {
16      "id": 2749908,
17      "full_name": "Andrew Mckenna",
18    }
19  ]
20}
```



Create > Add metadata > Publish



Create item

The screenshot shows the Postman application interface for creating an item. The top navigation bar displays "OR23 Workshop / Create item". The request method is set to "POST" and the URL is "https://api.figsh.com/v2/account/articles". The "Body" tab is selected, showing a JSON payload:

```
1 {  
2   ... "title": "My test item"  
3 }
```

The "Send" button is highlighted in blue at the top right. Below the body, the "Status: 201 Created" message is displayed along with performance metrics: Time: 149 ms, Size: 615 B. The "Pretty" view of the response JSON is shown:

```
1 {  
2   "entity_id": 8508592,  
3   "location": "https://api.figsh.com/v2/account/articles/8508592",  
4   "warnings": []  
5 }
```



Retrieve categories

The screenshot shows the Postman application interface. At the top, the URL is set to `https://api.figsh.com/v2/categories`. The 'Tests' tab is selected, containing the script:

```
1 pm.environment.set("categoryId", pm.response.json()[1].id);
```

The 'Body' tab is selected at the bottom, showing the JSON response:

```
12 {  
13   "is_selectable": true,  
14   "has_children": false,  
15   "id": 25484,  
16   "title": "Agricultural biotechnology diagnostics (incl. biosensors)",  
17   "parent_id": 25482,  
18   "path": "/25480/25482/25484",  
19   "source_id": "300101",  
20   "taxonomy_id": 2000  
21 },
```

The status bar at the bottom indicates: Status: 200 OK Time: 542 ms Size: 431.88 KB.



Update item

The screenshot shows the Postman application interface for making a PUT request to update an item. The URL is set to <https://api.figsh.com/v2/account/articles/8508592>. The 'Body' tab is selected, showing the JSON payload:

```
1 {
2   "title": "My test item",
3   "description": "Lorem ipsum dolor",
4   "categories": [{"categoryId"}],
5   "tags": ["test"]
6 }
```

The response status is 205 Reset Content, with a time of 233 ms and a size of 597 B. The response body is:

```
1 {
2   "location": "https://api.figsh.com/v2/account/articles/8508592",
3   "warnings": []
4 }
```



Publish item

The screenshot shows the Postman application interface for publishing an item. The top navigation bar includes 'OR23 Workshop / Publish item', 'Save' (with dropdown), three dots, a pencil icon, and a message icon. The main header shows 'POST https://api.figsh.com/v2/account/articles/8508592/publish' with a 'Send' button. Below the header, tabs for 'Params', 'Authorization' (which is selected and highlighted in orange), 'Headers (7)', 'Body', 'Pre-request Script', 'Tests', and 'Settings' are visible, along with a 'Cookies' tab. The 'Authorization' tab shows a 'Type' dropdown set to 'Bearer Token' and a 'Token' input field containing a long token string. A note below explains that the authorization header will be automatically generated. The 'Body' tab is selected, showing a JSON response with one object containing a 'location' key pointing to the published article's URL. The status bar at the bottom indicates a 'Status: 201 Created' response.

OR23 Workshop / Publish item

POST https://api.figsh.com/v2/account/articles/8508592/publish Send

Params Authorization ● Headers (7) Body Pre-request Script Tests Settings Cookies

Type Bearer Token Token c91a0bcfda12aef3a877508b64fafe5cb4e2...

The authorization header will be automatically generated when you send the request. Learn more about [authorization](#)

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON ↻

```
1 {  
2   "location": "https://api.figsh.com/v2/accounts/8508592"  
3 }
```

Status: 201 Created Time: 631 ms Size: 560 B Save as Example



Retrieve views

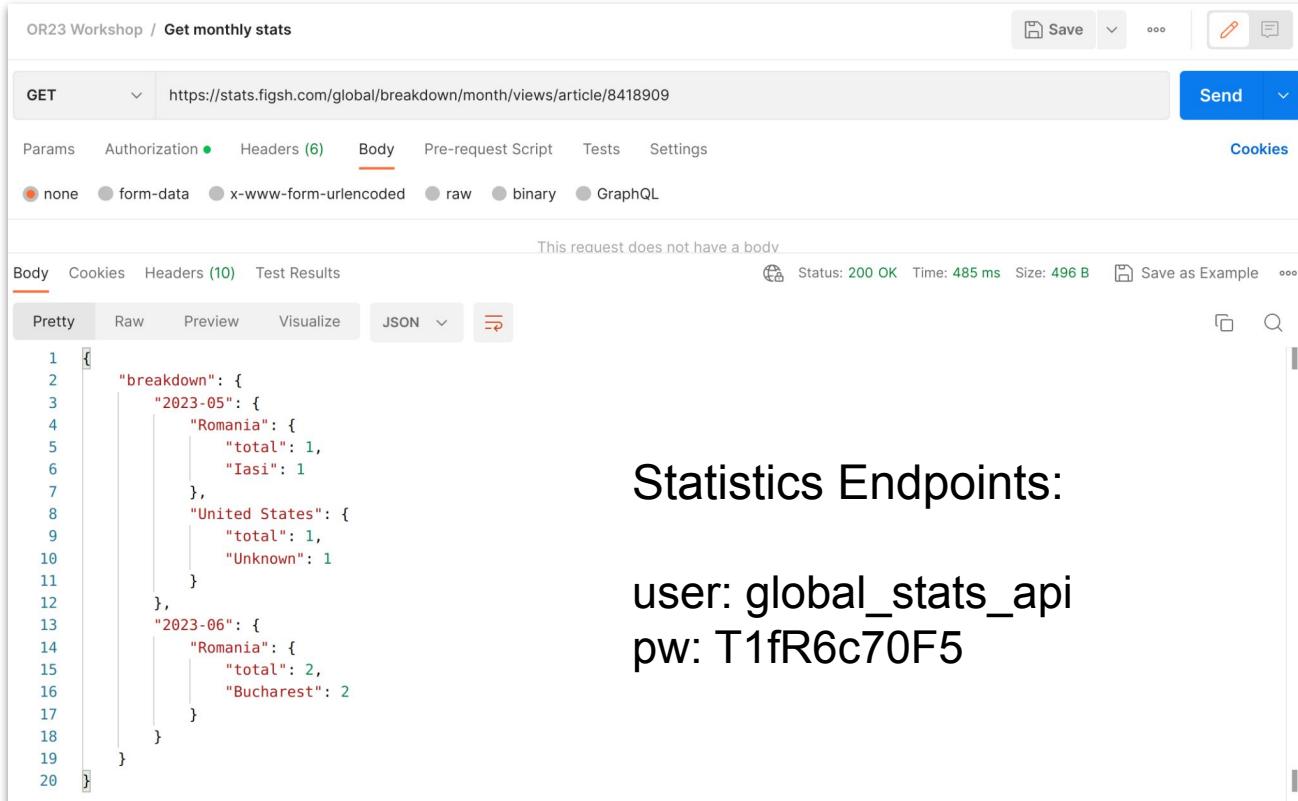
The screenshot shows the Postman application interface. At the top, a GET request is made to <https://stats.figsh.com/total/views/article/8460124>. The 'Authorization' tab is selected, showing 'Basic Auth' and 'global_stats' as the username. The 'Body' tab is selected, displaying a JSON response:

```
1 {  
2   "totals": 5  
3 }
```

At the bottom right, the status is shown as 200 OK with a time of 710 ms and a size of 291 B. There are also 'Save as Example' and other UI elements.



Retrieve monthly views breakdown



The screenshot shows the Postman application interface. The URL in the header is `https://stats.figsh.com/global/breakdown/month/views/article/8418909`. The 'Body' tab is selected, showing the response content:

```
1  [
2      "breakdown": {
3          "2023-05": {
4              "Romania": {
5                  "total": 1,
6                  "Iasi": 1
7              },
8              "United States": {
9                  "total": 1,
10                 "Unknown": 1
11             }
12         },
13         "2023-06": {
14             "Romania": {
15                 "total": 2,
16                 "Bucharest": 2
17             }
18         }
19     }
20 }
```

The status bar at the bottom indicates `Status: 200 OK Time: 485 ms Size: 496 B`.

Statistics Endpoints:

user: global_stats_api
pw: T1fR6c70F5



Retrieve repository accounts

The screenshot shows the Postman application interface. At the top, it displays the URL `https://api.figsh.com/v2/account/institution/accounts?limit=1&offset=0`. Below the URL, there are tabs for Params, Authorization, Headers (6), Body, Pre-request Script, Tests, and Settings. The Body tab is selected, showing the response body. The response body is a JSON object with the following structure:

```
1
2 {
3     "id": 1997142,
4     "first_name": "figshare admin",
5     "last_name": "global",
6     "email": "globaladmin@figsh.com.bk",
7     "active": 1,
8     "institution_id": 3682,
9     "institution_user_id": "",
10    "quota": 10737418240,
11    "used_quota": 159246054,
12    "user_id": 2750658
13 }
14 ]
```

The status bar at the bottom indicates a **200 OK** status, a **Time: 386 ms**, and a **Size: 934 B**.



Create account

The screenshot shows the Postman application interface for creating a new account. The top navigation bar includes 'OR23 Workshop / Create account', 'Save' (with a dropdown), three small icons, and a 'Send' button. Below the header, the method is set to 'POST' and the URL is 'https://api.figsh.com/v2/account/institution/accounts'. The 'Body' tab is selected, showing a JSON payload:

```
1 {
2   "email": "johndoe@example.com",
3   "first_name": "John",
4   "last_name": "Doe",
5   "group_id": 0,
6   "institution_user_id": "johndoe1",
7   "quota": 1000,
8   "is_active": true
9 }
```

Below the body, the status is shown as 'Status: 201 Created' with a timestamp of 'Time: 371 ms' and a size of 'Size: 473 B'. There are tabs for 'Body', 'Cookies', 'Headers (10)', and 'Test Results'. The 'Pretty' tab is selected in the preview section, displaying the response body:

```
1 {
2   "account_id": 2139246
3 }
```



PROJECT TIME

Postman

Use Postman to test out the endpoints you need for your project.

If your project requires chaining API calls, decide what language you will use. There are resources for [Python here](#).

Generate code using Postman

- Postman can generate source code in the language of your choice for the performed requests
- You can easily prototype using a *notebook*, for example
<https://colab.research.google.com/>



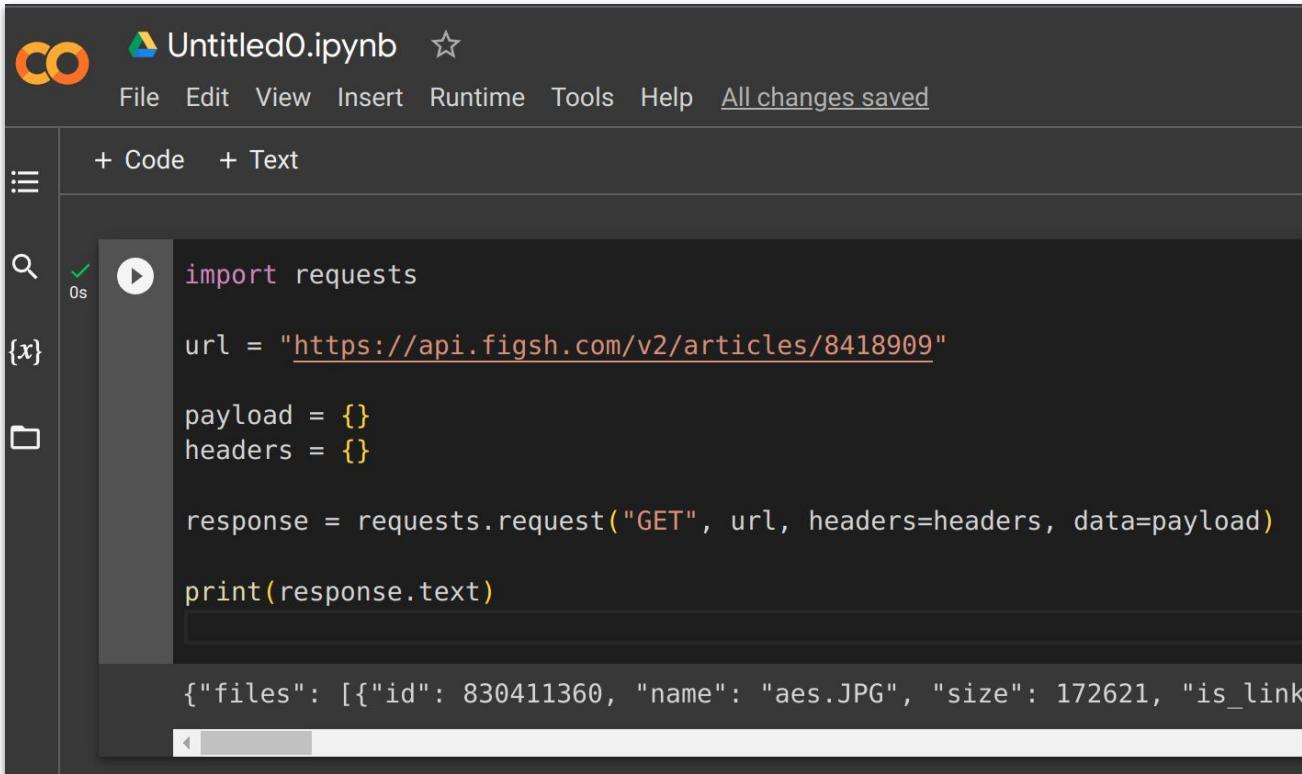
Generate code using Postman

The screenshot shows the Postman application interface. On the left, a request titled "Retrieve item metadata" is displayed. The method is GET, the URL is <https://api.figsh.com/v2/articles/8418909>, and the "Auth" tab is selected under "Params". A note states: "The authorization header will be automatically generated when you send this request". The "Headers" tab shows five entries. On the right, a "Code snippet" panel is open for "Python - Requests", containing the following code:

```
1 import requests
2
3 url = "https://api.figsh.com/v2/articles/
4     8418909"
5 payload = {}
6 headers = {}
7
8 response = requests.request("GET", url,
9     headers=headers, data=payload)
10
11 print(response.text)
```



Generate code using Postman



The screenshot shows a Jupyter Notebook interface with the title "Untitled0.ipynb". The code cell contains the following Python script:

```
import requests

url = "https://api.figsh.com/v2/articles/8418909"

payload = {}
headers = {}

response = requests.request("GET", url, headers=headers, data=payload)

print(response.text)
```

The output of the code is displayed below the cell:

```
{"files": [{"id": 830411360, "name": "aes.JPG", "size": 172621, "is_link": false}]}  
[{"id": 830411360, "name": "aes.JPG", "size": 172621, "is_link": false}]
```



Tea Break
30 min





Building an application

Requirements

- Retrieve the items from <https://global.figsh.com>
- Retrieve the number of views for each item
- Display the data on a HTML page



Tools

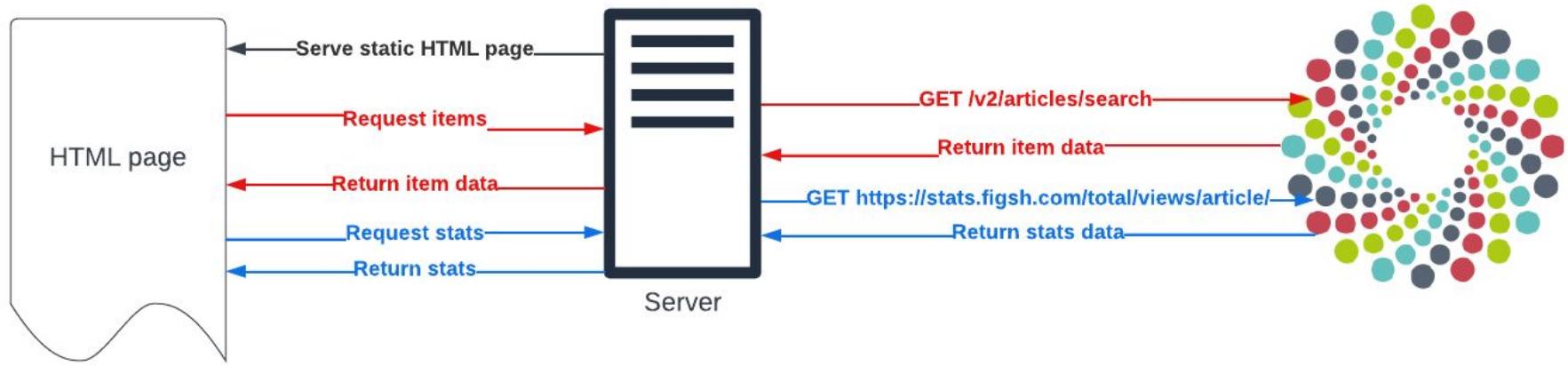
- Server:
 - Python: <https://www.python.org/>
 - Flask: <https://flask.palletsprojects.com/en/2.3.x/>
- Web interface:
 - React: <https://react.dev/>
- Development environment:
 - Visual Studio Code: <https://code.visualstudio.com/>
 - Python in Visual Studio Code:
<https://code.visualstudio.com/docs/python/python-tutorial>
- Code generation:
 - ChatGPT: <https://chat.openai.com/>

CORS

- Abbreviation for cross-origin resource sharing
- Mechanism which indicates from what origins (<http://localhost:5000> in our case) a browser is allowed to make HTTP API requests
- Security feature which prevents malicious sites stealing user information from legitimate sites
- By default Figshare's API does not allow cross-origin requests; solutions:
 - Make API requests via a (proxy) server
 - Contact support@figshare.com



System diagram



Further documentation

- Python Flask: <https://flask.palletsprojects.com/en/2.3.x/>
- Python requests: <https://requests.readthedocs.io/en/latest/>
- React: <https://react.dev/>
- Fetch: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API
- Promise:
https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Promise



And that's everything

- What we've covered:
 - Basics of HTTP APIs
 - Overview of Figshare's APIs
 - How to use Postman to perform HTTP requests
 - How to prototype applications that make use of an API
 - How to build a basic application that displays data from Figshare's API



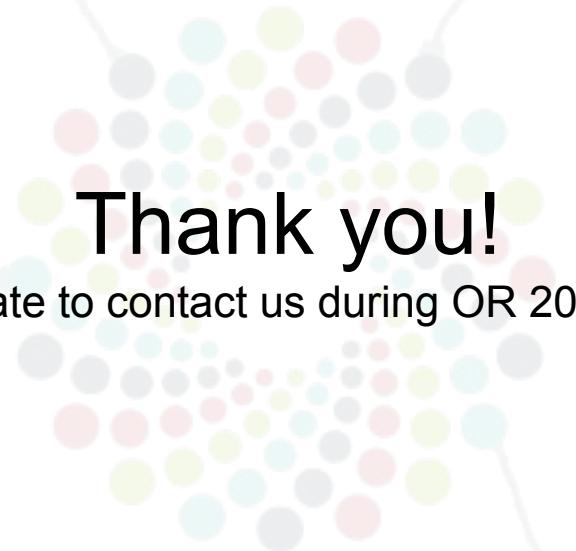
And that's everything

- What we'll make available to you:
 - This presentation:
<https://docs.google.com/presentation/d/1fTqMwE5c2zuip0B7I7oQgxvAjVklIBb1-YTqEYx8XQ/edit?usp=sharing>
 - The Postman collection with the performed requests:
<https://www.postman.com/tudor/workspace/public-workspace/collection/329625-1a755e1d-811a-48b8-8471-860e3f09f0ec>
 - The ChatGPT chat:
<https://chat.openai.com/share/fea4383b-3b32-4608-9699-69aca739aa6f>
 - The workshop website:
<https://amckennafoster.github.io/figshare-api-workshop.github.io/index.html>



Q&A & PROJECT TIME

Use this time to ask questions and work on your project.



Thank you!

And don't hesitate to contact us during OR 2023 and beyond!