

# Figshare API Workshop

Open Repositories 2023  
Stellenbosch, South Africa  
Twitter: #openrepos2023



# Quick Activity



Photo by [iorni.com](https://www.iorni.com) on [Unsplash](https://www.unsplash.com)



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

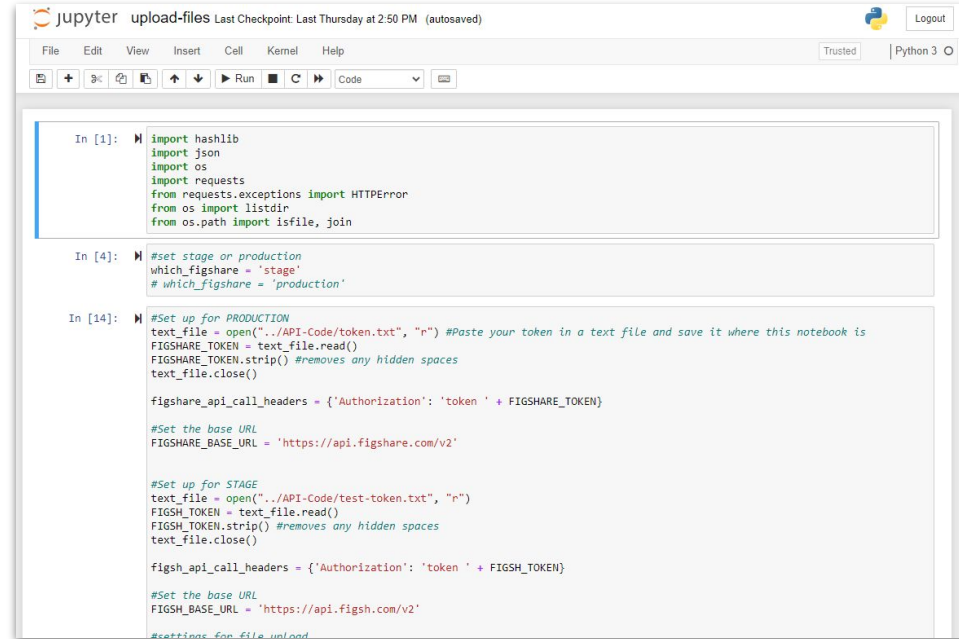
#Gather basic metadata for items (articles) that meet your search criteria
query = f'{BASE_URL}/items?author={name}&page_size=100' #Get up to 100
```



# Examples

## Manage repository records

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



The screenshot shows a Jupyter Notebook window titled "upload-files" with a last checkpoint from Thursday at 2:50 PM. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations and execution. The notebook contains three code 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_figshare = 'stage'
# which_figshare = '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 institutional repository.
- The existing (dryad-generated) Figshare record is updated with the generated Figshare record.
- The 'link file' option is used on the command line to link the file to the record.
- This is 'one-time only' script - the script is not intended to be an ongoing harvesting tool.

### Prerequisites

To use this script, you will need to create a Figshare repository account under which records will be deposited.

```
# SET_TOKEN
```

Macquarie University

Browse Search on Macquarie University Log in

File(s) stored somewhere else

<https://doi.org/10.5061/dryad.ct121>

Please note: Linked content is NOT stored on Macquarie University and we can't guarantee its availability, quality, security or accept any liability.

**Data from: Developmental stress increases reproductive success in male zebra finches**

Cite Share Collect

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

USAGE METRICS

19 views	0 downloads	0 citations
----------	-------------	-------------

There is increasing evidence that exposure to stress during development can have

[https://github.com/mq-eresearch/dryad\\_to\\_figshare/tree/v1.0.0](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 displays the ORDA website interface. At the top, there is a navigation bar with the University of Sheffield logo, the ORDA logo, and links for Home, Browse ORDA, Visualisation Showcase, Upload, Help, Feedback, and a search icon. Below the navigation bar, there is a search bar with the text "Discover our research." and a search input field containing "All fields" and "data". To the right of the search bar, there are filters for "Filter: All types" and "Sort: Date published". Below the search bar, there are several search results displayed as cards. Each card includes a title, a brief description, the author's name, the date of publication, and icons for various file formats (PDF, HTML, etc.). The interface is clean and modern, with a navigation menu on the left and a search bar at the top.





# 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

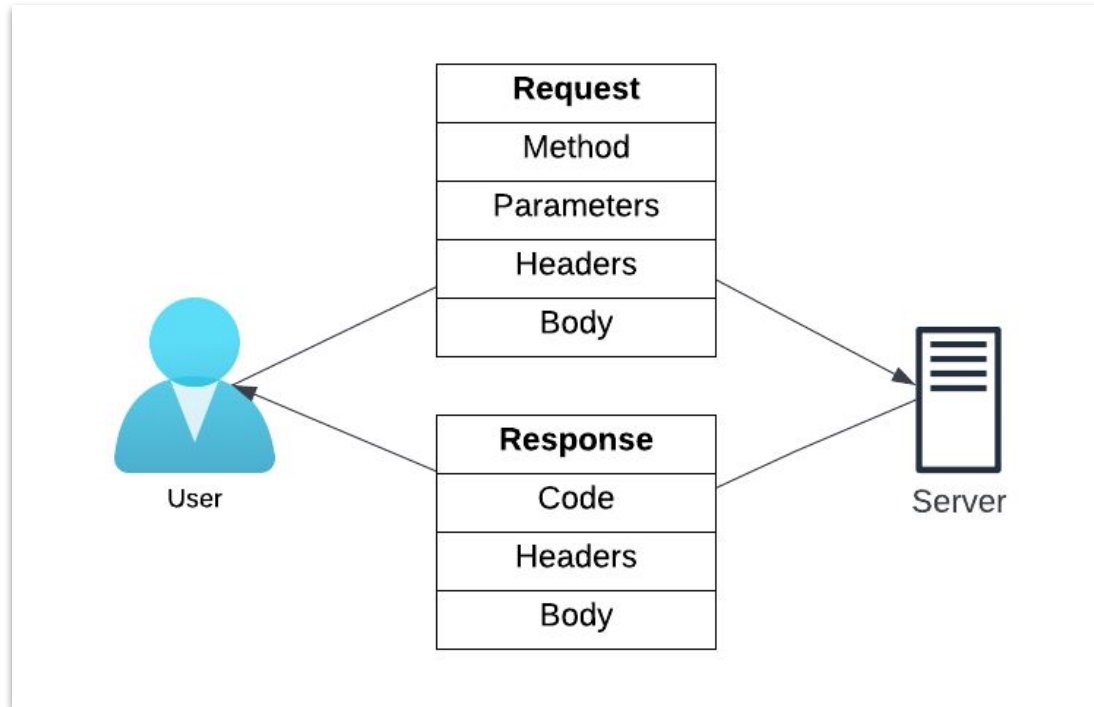
- Introduction to HTTP requests and APIs
- Introduction to Figshare's API(s)
- Accessing Figshare's REST API using Postman
- Building an application over Figshare's REST API





# 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



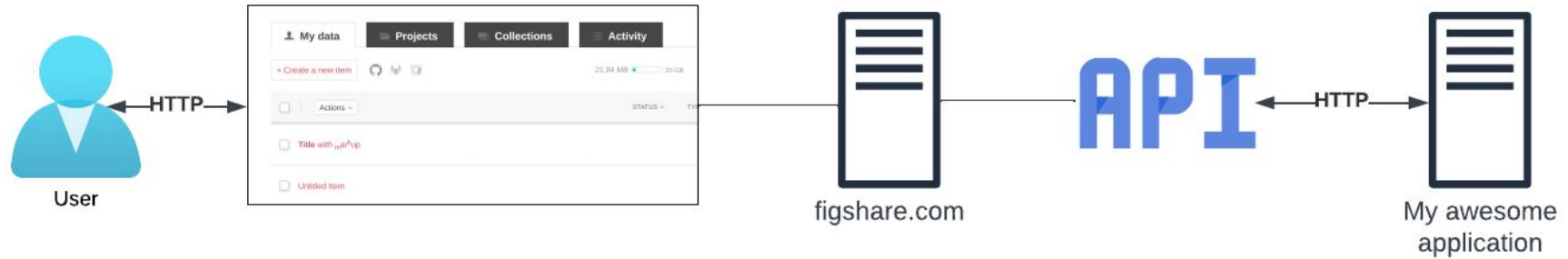
# What is an API?

- 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": "87f39b4da1a371572523b59bf83bb32e",
      "computed_md5": "87f39b4da1a371572523b59bf83bb32e"
    }
  ],
  "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](https://github.com/figshare/user_documentation)
- Swagger file:  
[https://github.com/figshare/user\\_documentation/blob/master/swagger\\_documentation/swagger.json](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](mailto: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

<b>global.user@figsh.com.bk</b>
Bongani Jwara
Charl Roberts
Nambitha Manqola
Nkululeko Magwaza

<b>global.user2@figsh.com.bk</b>
Andisiwe Magocoba
Bubele Bido
Janina Van der Westhuizen
Yan Han

<b>global.user3@figsh.com.bk</b>
Samuel Simango
Songezo Mpikashe
Sizwe Ngcobo

<b>global.user4@figsh.com.bk</b>
Regina Sikhosana
Richard Nobebe
Xabiso Xesi

<b>global.user5@figsh.com.bk</b>
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

OR23 Workshop / Retrieve item metadata

GET <https://api.figsh.com/v2/articles/8418909> 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 (12) Test Results Status: 200 OK Time: 448 ms Size: 4.04 KB Save as Example

Pretty Raw Preview Visualize JSON

```
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",
```



Create > Add metadata > Publish



# Create item

The screenshot displays a REST client interface for a 'Create item' endpoint. The request is a POST to `https://api.figsh.com/v2/account/articles` with a JSON body: `{ "title": "My test item" }`. The response is a 201 Created status with a response time of 149 ms and a size of 615 B. The response body is shown in JSON format: `{ "entity_id": 8508592, "location": "https://api.figsh.com/v2/account/articles/8508592", "warnings": [] }`.

OR23 Workshop / Create item

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

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies

none  form-data  x-www-form-urlencoded  raw  binary  GraphQL **JSON** Beautiful

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

**Body** Cookies Headers (11) Test Results Status: 201 Created Time: 149 ms Size: 615 B Save as Example

Pretty Raw Preview Visualize **JSON** 🔍

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



# Retrieve categories

The screenshot shows a REST client interface with the following details:

- URL:** `https://api.figsh.com/v2/categories`
- Method:** GET
- Params:** None
- Authorization:** None
- Headers:** 6
- Body:** None
- Pre-request Script:** None
- Tests:** 1 test script: `pm.environment.set("categoryId", pm.response.json()[1].id);`
- Settings:** None
- Response:** Status: 200 OK, Time: 542 ms, Size: 431.88 KB
- Response Body (JSON):**

```
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 },
```



# Update item

The screenshot shows a REST client interface for an API. The title bar indicates the current workspace is 'OR23 Workshop / Update item'. The main area is set to a 'PUT' method for the URL 'https://api.figsh.com/v2/account/articles/8508592'. The 'Body' tab is selected, showing a JSON payload with fields for title, description, categories, and tags. The response area at the bottom shows a 205 status code and a JSON response with location and warnings fields.

OR23 Workshop / Update item

PUT <https://api.figsh.com/v2/account/articles/8508592> Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded **raw** binary GraphQL JSON

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

Body Cookies Headers (11) Test Results Status: 205 Reset Content Time: 233 ms Size: 597 B Save as Example

Pretty Raw Preview Visualize JSON

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



# Publish item

The screenshot shows a REST client interface for a "Publish item" endpoint. The request is a POST to `https://api.figsh.com/v2/account/articles/8508592/publish`. The "Authorization" tab is active, showing a Bearer Token of `c91a0bcfda12aef3a877508b64fafa5cb4e2!...`. The response body is shown in JSON format, containing a `location` property with the value `https://api.figsh.com/v2/articles/8508592`. The status is 201 Created, with a time of 631 ms and a size of 560 B.

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: `c91a0bcfda12aef3a877508b64fafa5cb4e2!...`

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

Body Cookies Headers (11) Test Results Status: 201 Created Time: 631 ms Size: 560 B Save as Example

Pretty Raw Preview Visualize JSON

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



# Retrieve views

The screenshot displays a REST client interface for a GET request to `https://stats.figshare.com/total/views/article/8460124`. The request is configured with Basic Authentication, where the username is `global_stats`. The response status is `200 OK` with a response time of `710 ms` and a size of `291 B`. The response body is shown in JSON format as `{ "totals": 5 }`.

GET `https://stats.figshare.com/total/views/article/8460124` Send

Params Authorization Headers (6) Body Pre-request Script Tests Settings Cookies

Type Basic Auth Username `global_stats`

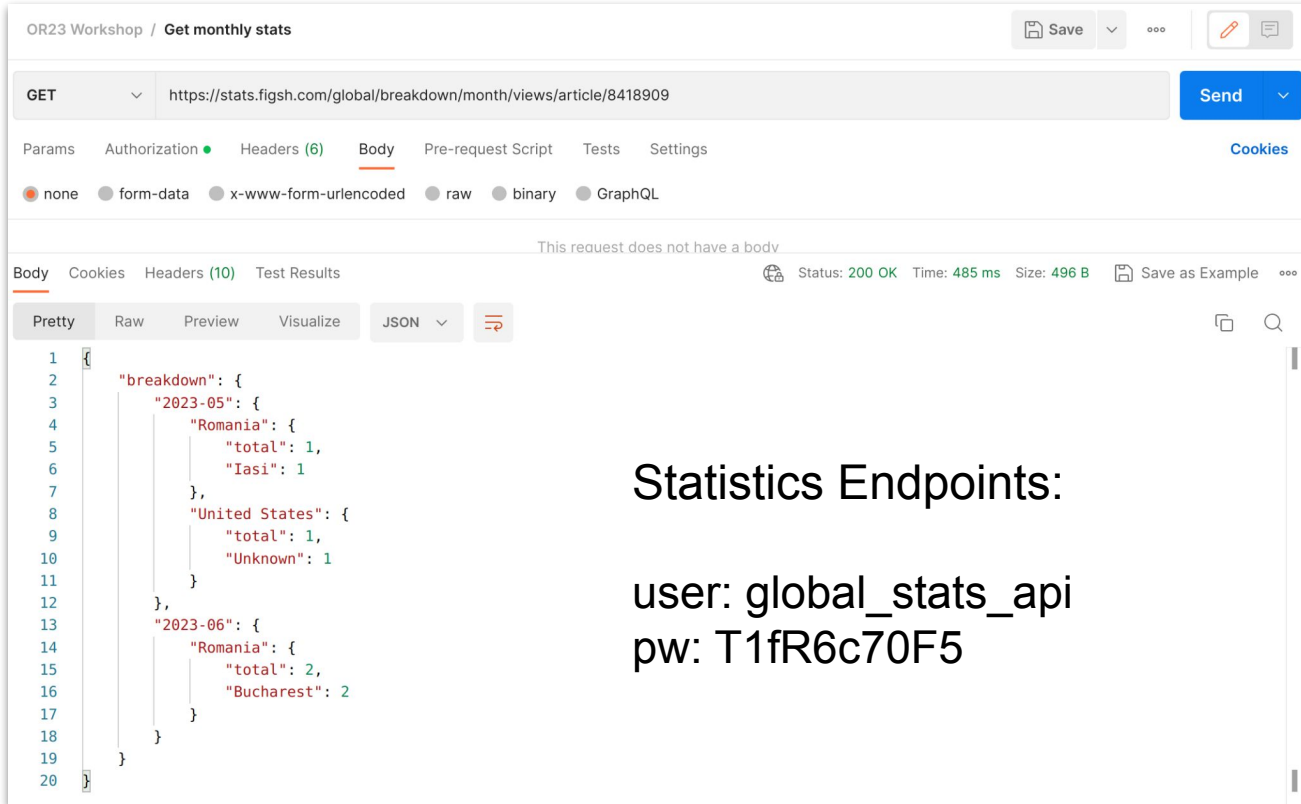
Body Cookies Headers (8) Test Results Status: 200 OK Time: 710 ms Size: 291 B Save as Example

Pretty Raw Preview Visualize JSON

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



# Retrieve monthly views breakdown



OR23 Workshop / Get monthly stats

GET <https://stats.figsh.com/global/breakdown/month/views/article/8418909> Send

Params Authorization Headers (6) **Body** Pre-request Script Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

Body Cookies Headers (10) Test Results Status: 200 OK Time: 485 ms Size: 496 B Save as Example

Pretty Raw Preview Visualize JSON

```
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 }
```

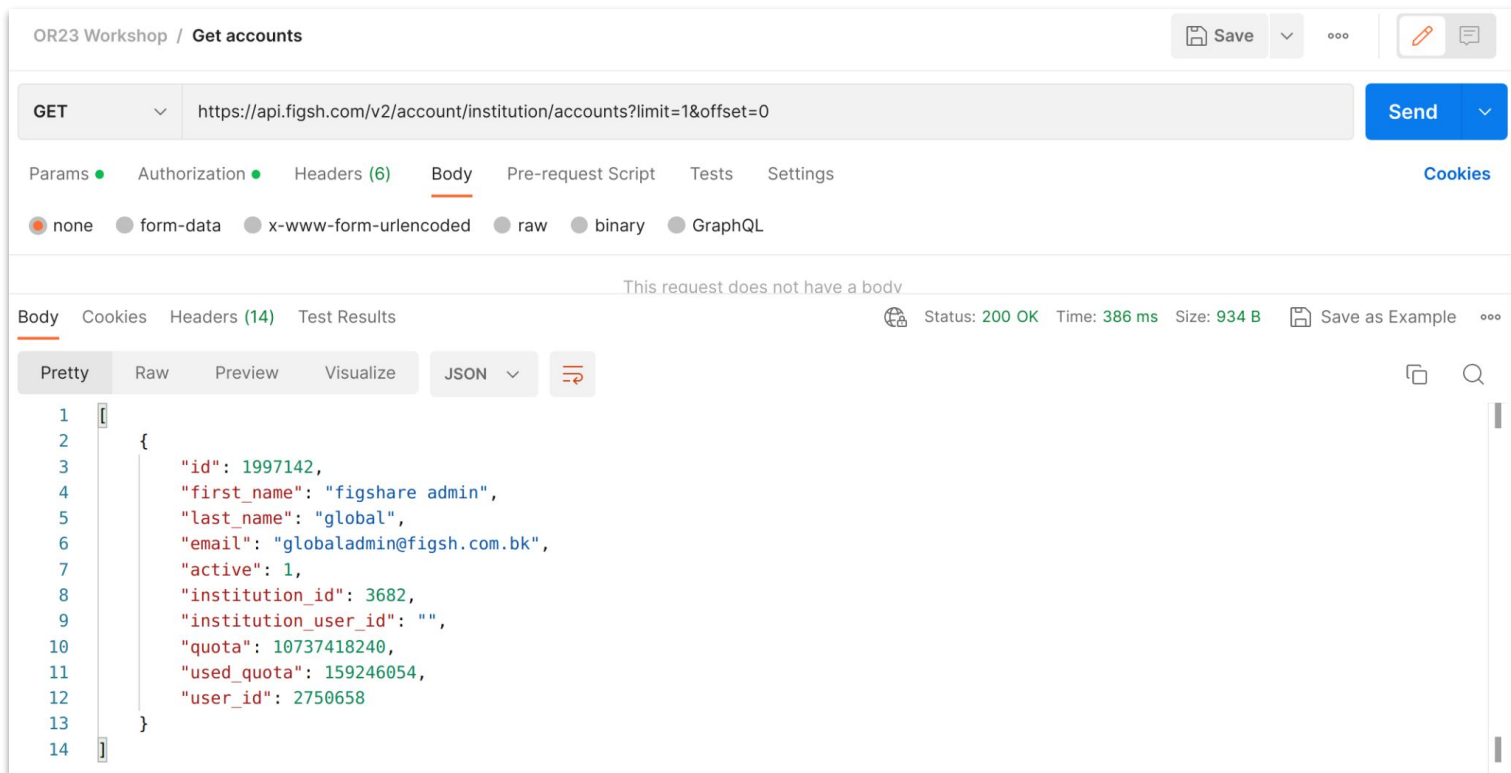
Statistics Endpoints:

user: global\_stats\_api  
pw: T1fR6c70F5





# Retrieve repository accounts



The screenshot shows a REST client interface for a request to `https://api.figsh.com/v2/account/institution/accounts?limit=1&offset=0`. The response is a JSON object representing a repository account.

```
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 ]
```



# Create account

The screenshot displays a REST client interface for a POST request to the endpoint `https://api.figsh.com/v2/account/institution/accounts`. The request body is a JSON object with the following fields: `email` (johndoe1@example.com), `first_name` (John), `last_name` (Doe), `group_id` (0), `institution_user_id` (johndoe1), `quota` (1000), and `is_active` (true). The response is a JSON object with the field `account_id` (2139246). The status is 201 Created, with a response time of 371 ms and a size of 473 B.

OR23 Workshop / Create account

POST `https://api.figsh.com/v2/account/institution/accounts` Send

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies Beautify

none form-data x-www-form-urlencoded **raw** binary GraphQL **JSON**

```
1 {
2   "email": "johndoe1@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 }
```

Body Cookies Headers (10) Test Results Status: 201 Created Time: 371 ms Size: 473 B Save as Example

Pretty Raw Preview Visualize **JSON**

```
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/>



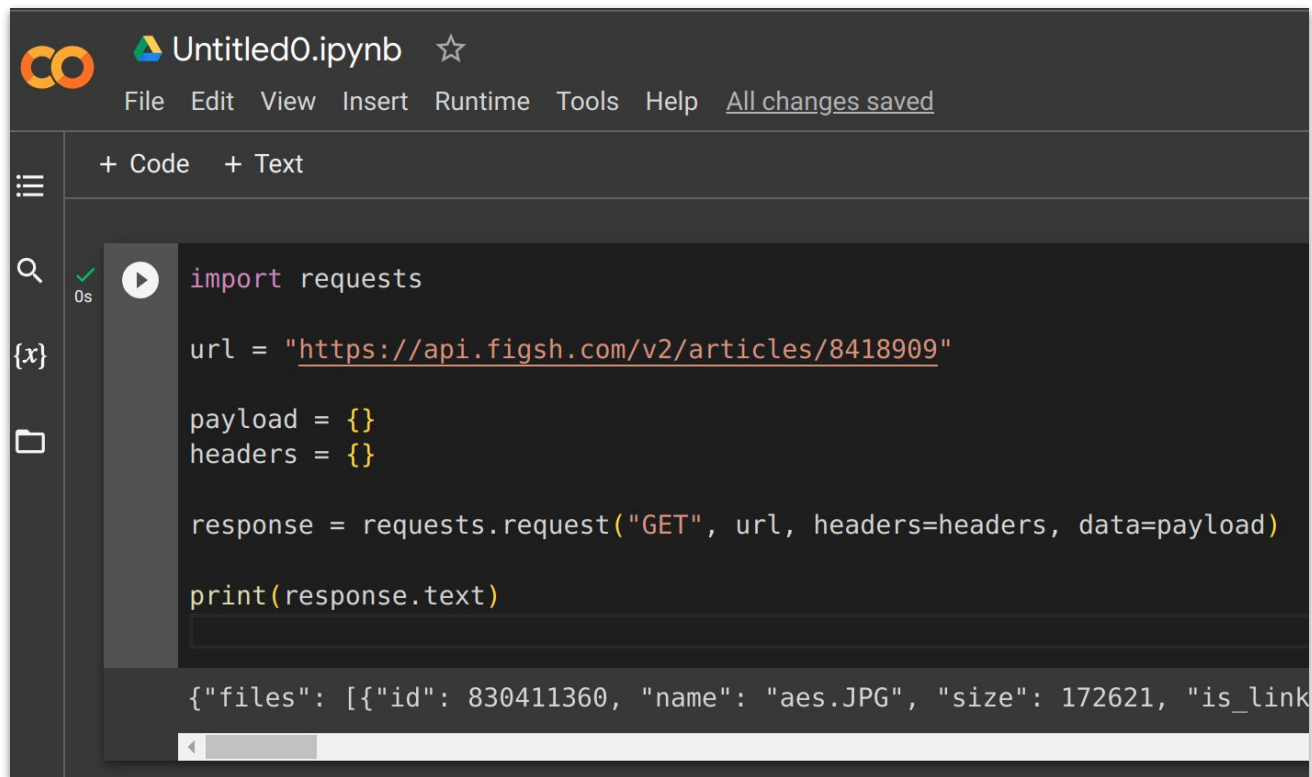
# Generate code using Postman

The screenshot displays the Postman interface for a GET request. The request is named "Retrieve item metadata" and is part of a collection "OR23 Workshop". The URL is "https://api.figsh.com/v2/articles/8418909". The "Auth" tab is selected, showing "Inherit auth from p..." and a note: "The authorization header will be automatically generated when you send". Below this, it states "This request is using No Auth from collection OR23 Workshop." The "Response" tab is also visible. On the right, the "Code snippet" panel shows the generated Python code:

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



# Generate code using Postman



```
import requests

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

payload = {}
headers = {}

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

print(response.text)

{"files": [{"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/>

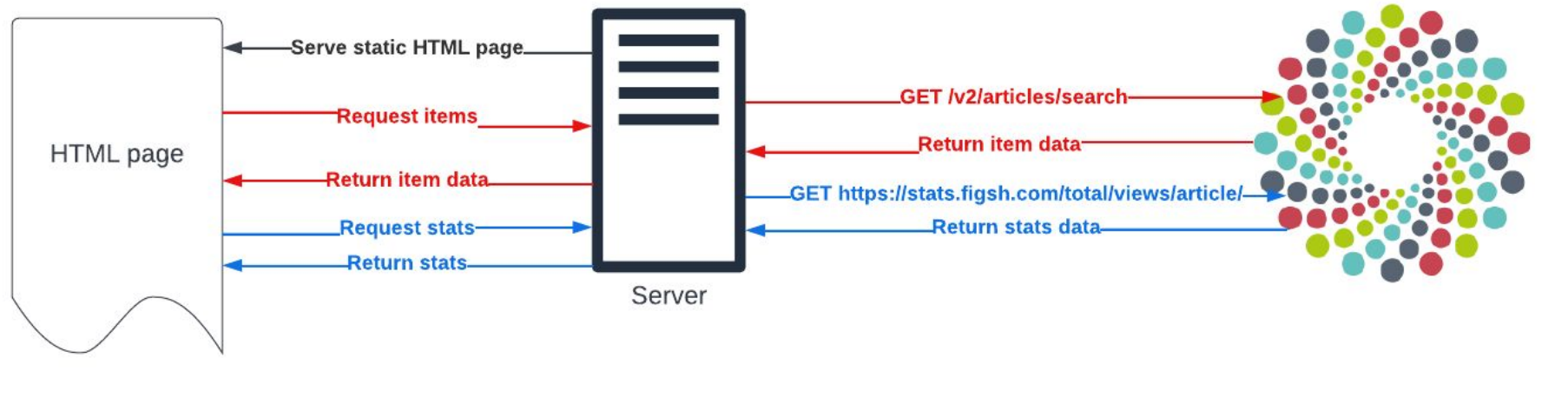


# CORS

- 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](mailto: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](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](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://amckennafoster.github.io/figshare-api-workshop.github.io/assets/Figshare-API-Workshop-Open-Repositories-2023.pdf>
  - The Postman collection with the performed requests:  
<https://www.postman.com/tudor/workspace/public-workspace/collection/329625-1a755e1d-811a-48b8-8471-860e3f09f0ec>
  - The ChatGPT chats:
    - Workshop:  
<https://chat.openai.com/share/28e38008-350f-420f-b53a-9710d9987817> (or PDF)
    - Complete:  
<https://chat.openai.com/share/fea4383b-3b32-4608-9699-69aca739aa6f> (or PDF)
  - The workshop website:  
<https://amckennafoster.github.io/figshare-api-workshop.github.io/index.html>



The background features a central cluster of multi-colored dots (green, yellow, orange, grey) arranged in a roughly circular pattern. Four light grey cables with connectors extend from the corners towards the center, overlapping the dot cluster.

# **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!