

# DEVELOPING AN ANALYSIS SOFTWARE'S API EXTENSION

Xael Shan

Poolesville Highschool

Mentor - Jeff Krzywon, Elizabeth Kelley

# SASVIEW OVERVIEW

## What is SANS?

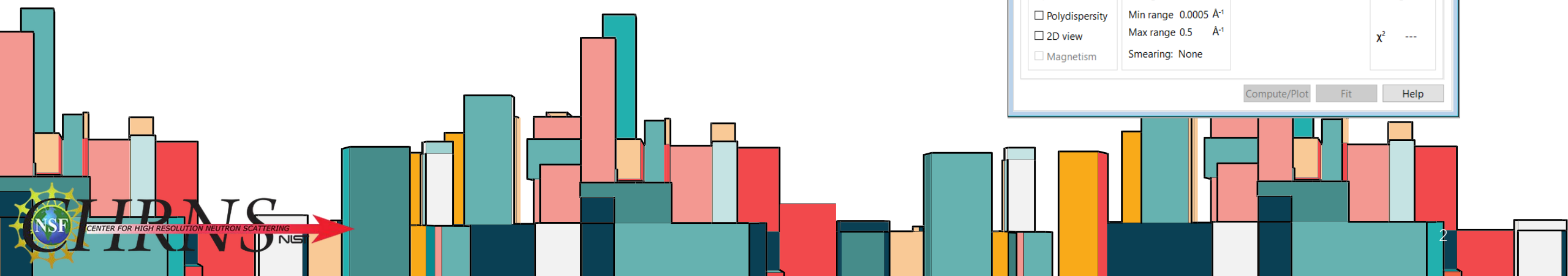
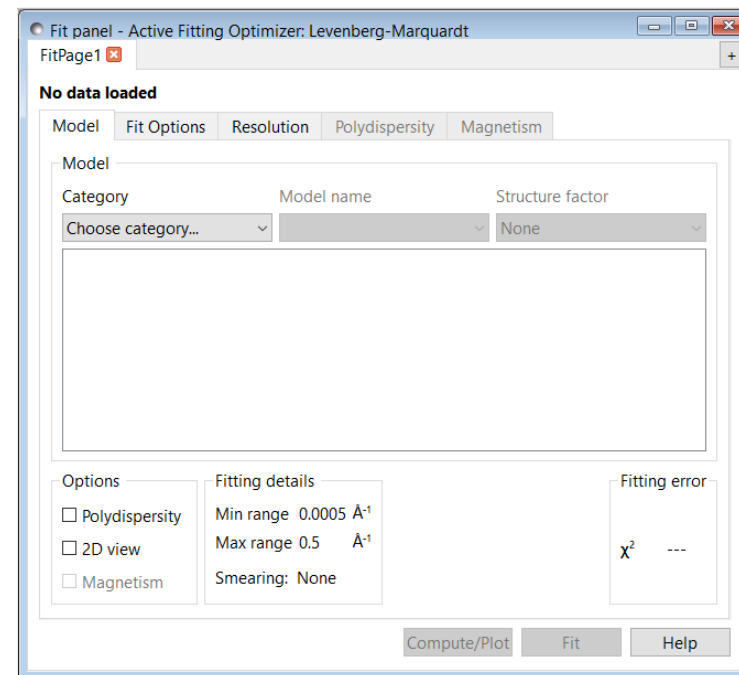
NCNR is a user facility that uses neutron scattering: Small Angle Neutron Scattering (SANS)

- Shoots neutron through a sample

## What is SasView?

SasView is an analysis software that takes the data from SANS:

- Analysis Tools
  - Fit
- SasCalc Tools



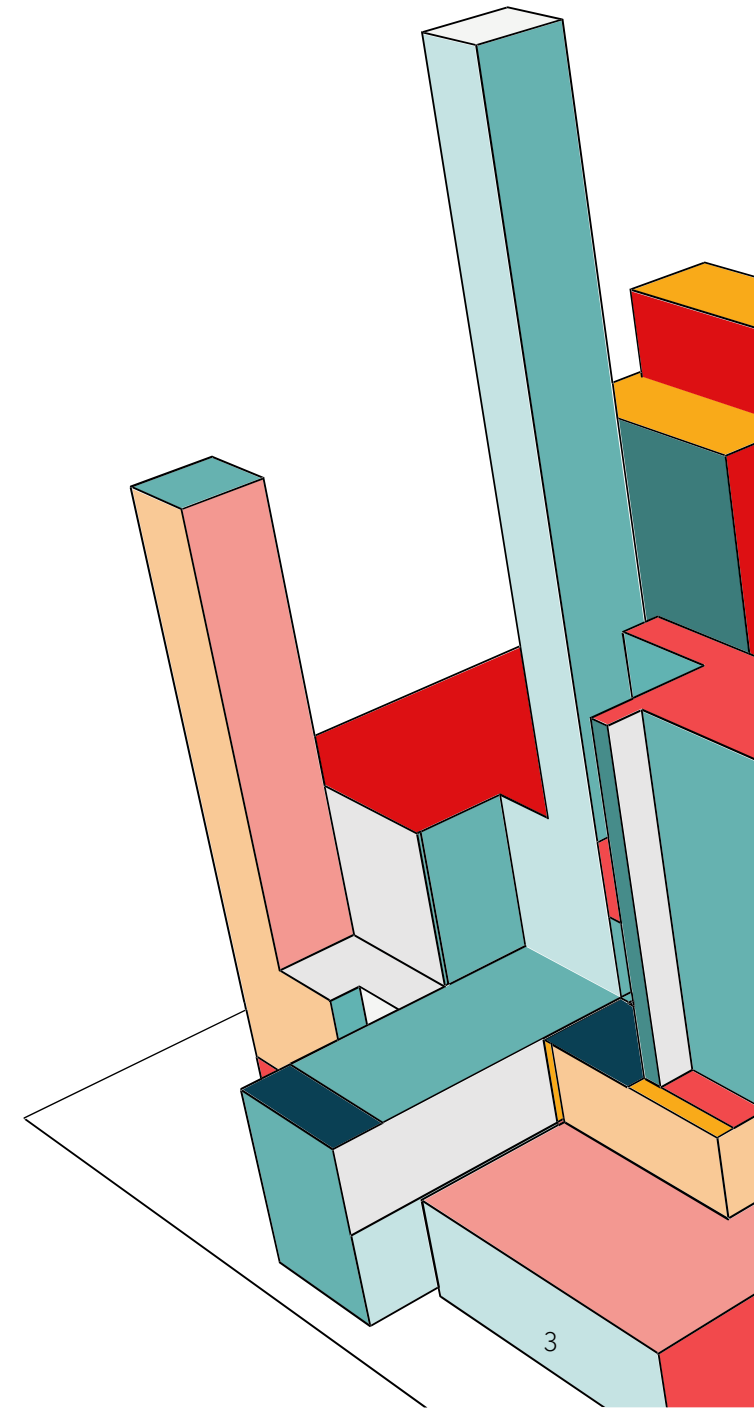
# PROJECT OVERVIEW

## The Project

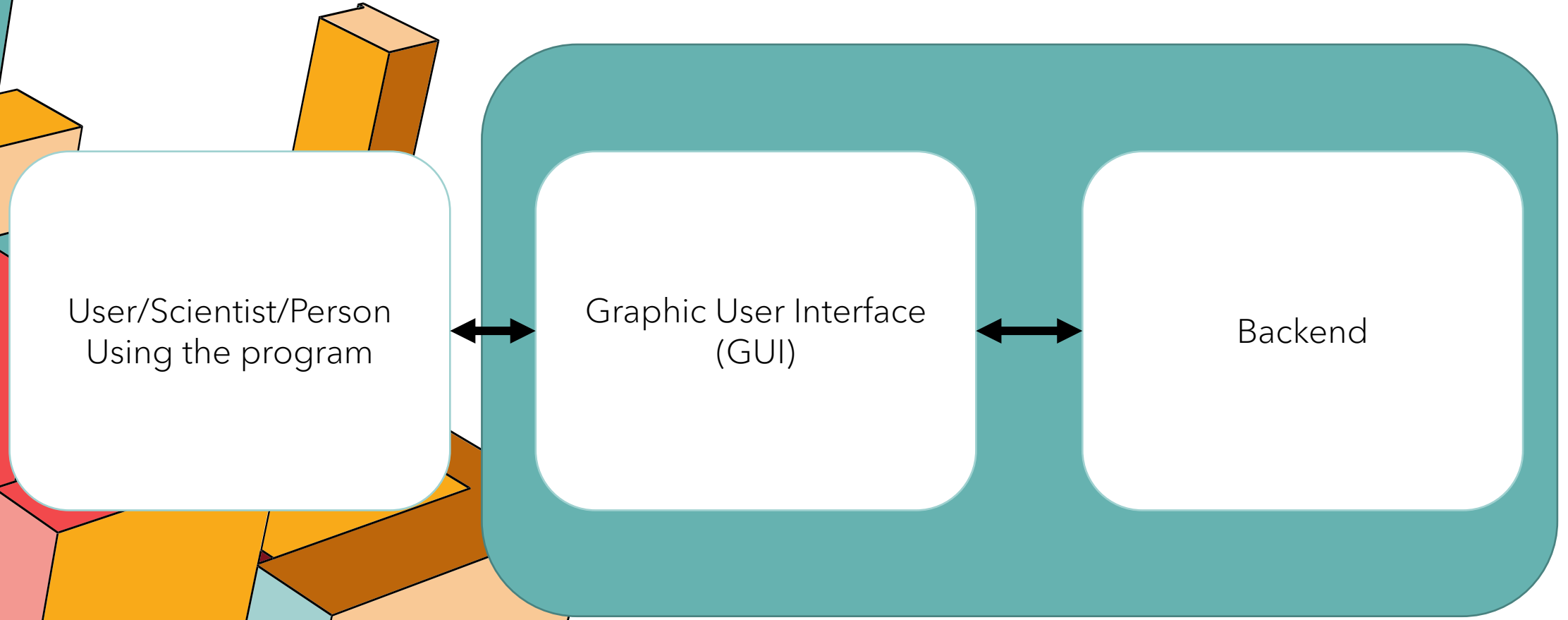
Create a website-based application to give more control and flexibility to extend Sasview functions

## Goals

- Create a working model of a website-based application
- Use application to analyze bicelle data
- Future Publish API

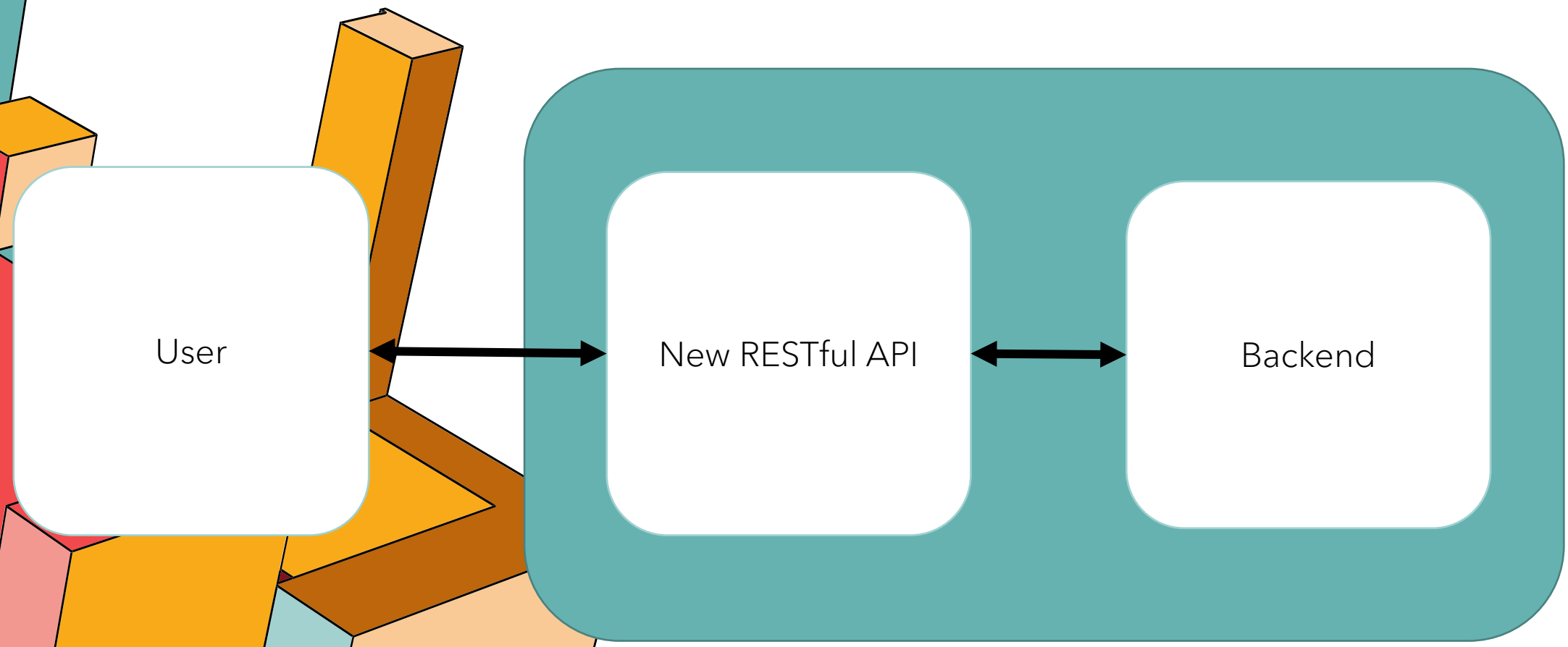


# Current Layout



Sasview/Local Computer

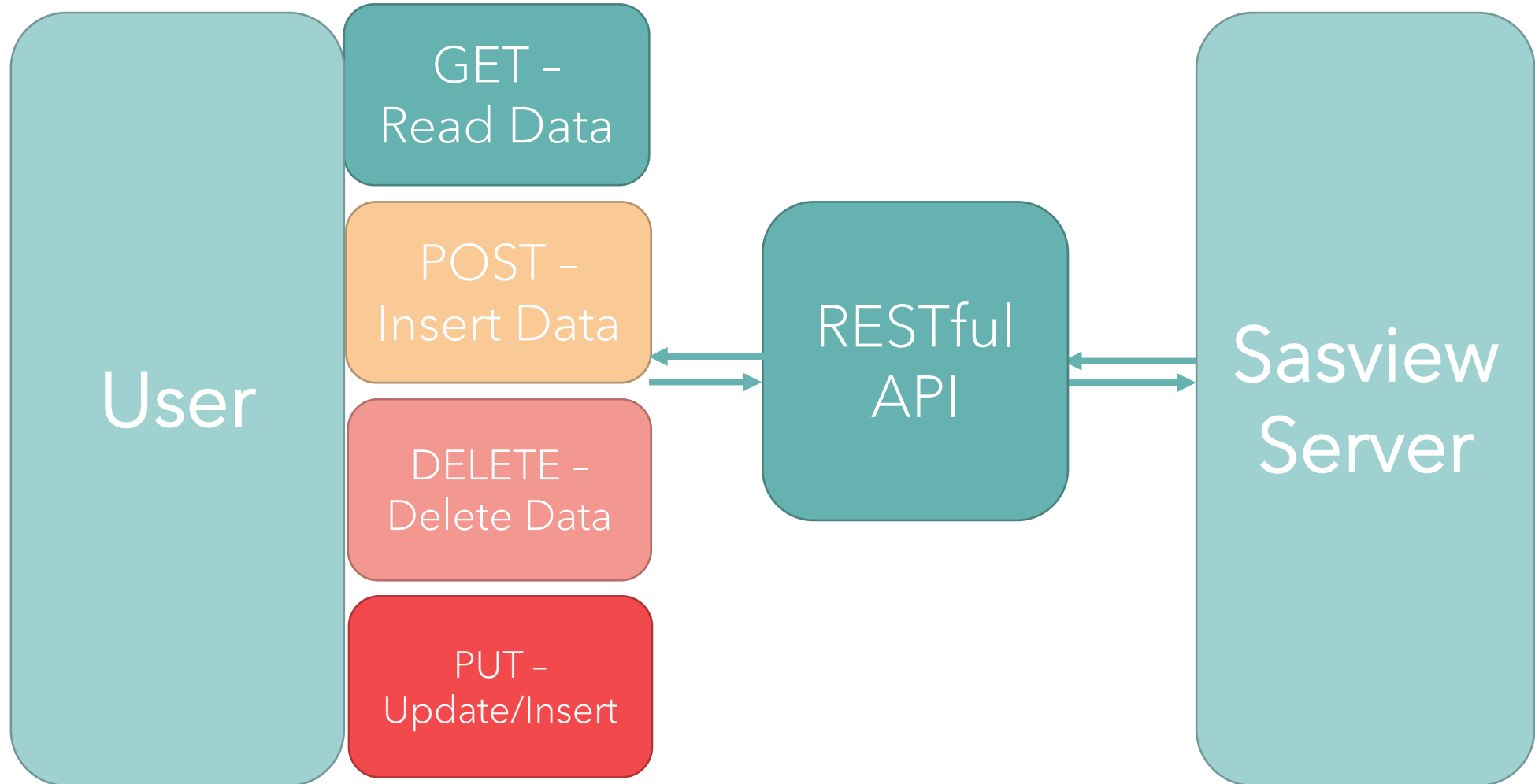
# Proposed Layout



Remote

# WHAT IS AN API

RESTful Framework Inputs



# WHY DO MORE?

Why create "another" Sasview

**Accessible**

Alleviate Resources

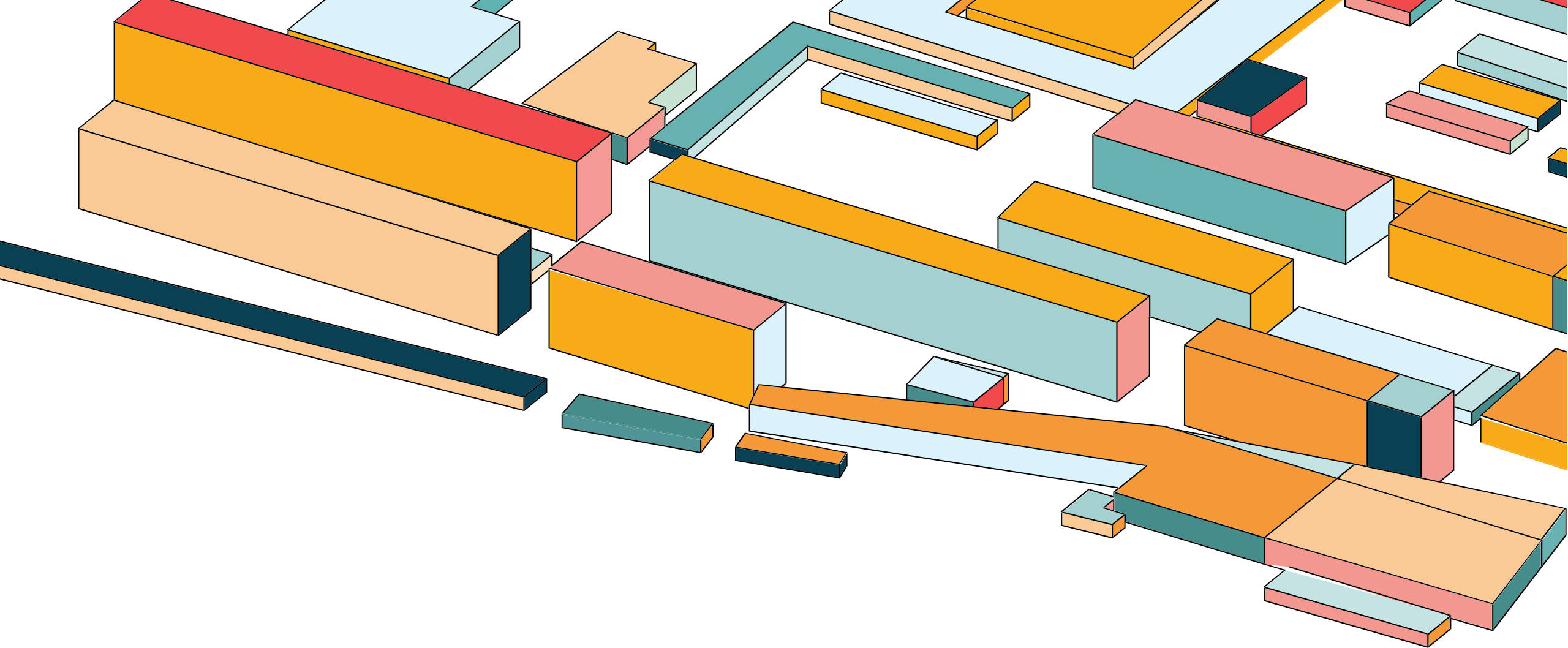
**Experimental  
benefits**

Non-Equilibrium  
(Structure of Materials)  
Initiative

Automation!

**Operating  
System  
Deployment**

Overcomes Window,  
Linux, Mac  
Requirements



# MAKING THE API

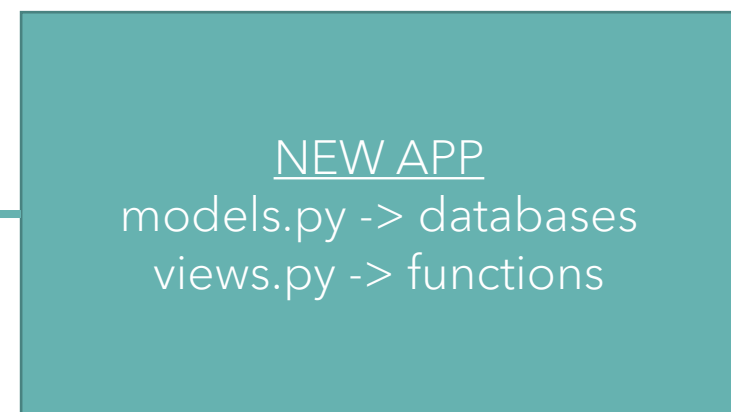
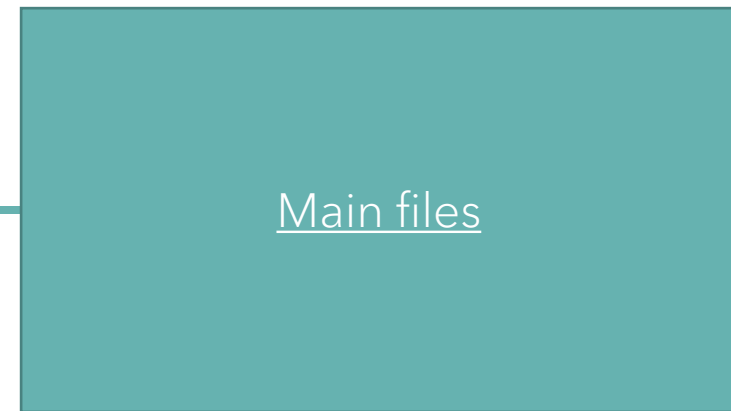


# DJANGO

Modularity!

```
▼ webfit
  ▼ analyze
    > __pycache__
  ▼ fitting
    > __pycache__
    > migrations
    🌀 __init__.py
    🌀 admin.py
    🌀 apps.py
    🌀 models.py
    🌀 tests.py
    🌀 views.py
    🌀 models.py
    🌀 urls.py
  ▼ data
    > __pycache__
    > migrations
    🌀 __init__.py
    🌀 admin.py
    🌀 apps.py
    🌀 models.py
    🌀 tests.py
    🌀 urls.py
    🌀 views.py
```

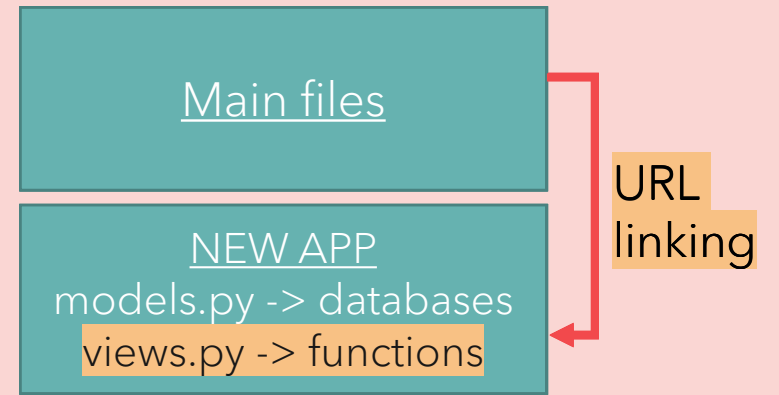
```
▼ user_authentication
  > __pycache__
  > migrations
  > views
  🌀 __init__.py
  🌀 admin.py
  🌀 apps.py
  🌀 models.py
  🌀 tests.py
  🌀 urls.py
  ▼ webfit
    > __pycache__
    🌀 __init__.py
    🌀 asgi.py
    🌀 settings.py
    🌀 urls.py
    🌀 versioning.py
    🌀 wsgi.py
  🌀 db.sqlite3
  🌀 manage.py
```



URL linking

# WHAT I IMPLEMENTED

How this was made from scratch



### Urls

- Links everything together!
- Allows the user to get to function

Website.com/homepage/  
Website.com/homepage/login

### Models.py

- Database structure
- Define what the user inputs
- Make sure user can't put in malicious data

### Views.py

- Translator
- Puts into databases
- Defines how user's data looks

Data id	user	file	file_name	is_public	file_name	is_public
1	None	<File_obj>	Data.txt	True	Inputted data	True
2	None	<File_obj>	Hello.txt	False		
3...	None					

TRANSLATE

“is\_public” : False  
“file”:file\_obj  
“file\_name” : “hello.txt”

127.0.0.1:8000/v1/data/upload/?data\_id=1

# WHERE ARE WE NOW!

**1**

Database

Created!

**2**

Views

Written!

**3**

Let's show it  
working!

The working model

# MULTIPLE WAYS TO BE A USER

Django REST framework

List Data

OPTIONS GET

GET /v1/data/list/

HTTP 200 OK  
Allow: OPTIONS, GET  
Content-Type: application/json  
Vary: Accept

```
{
  "public_file_ids": [
    {
      "1": "10000A_sphere_dsm.xml"
    }
  ]
}
```

Development Server: API view

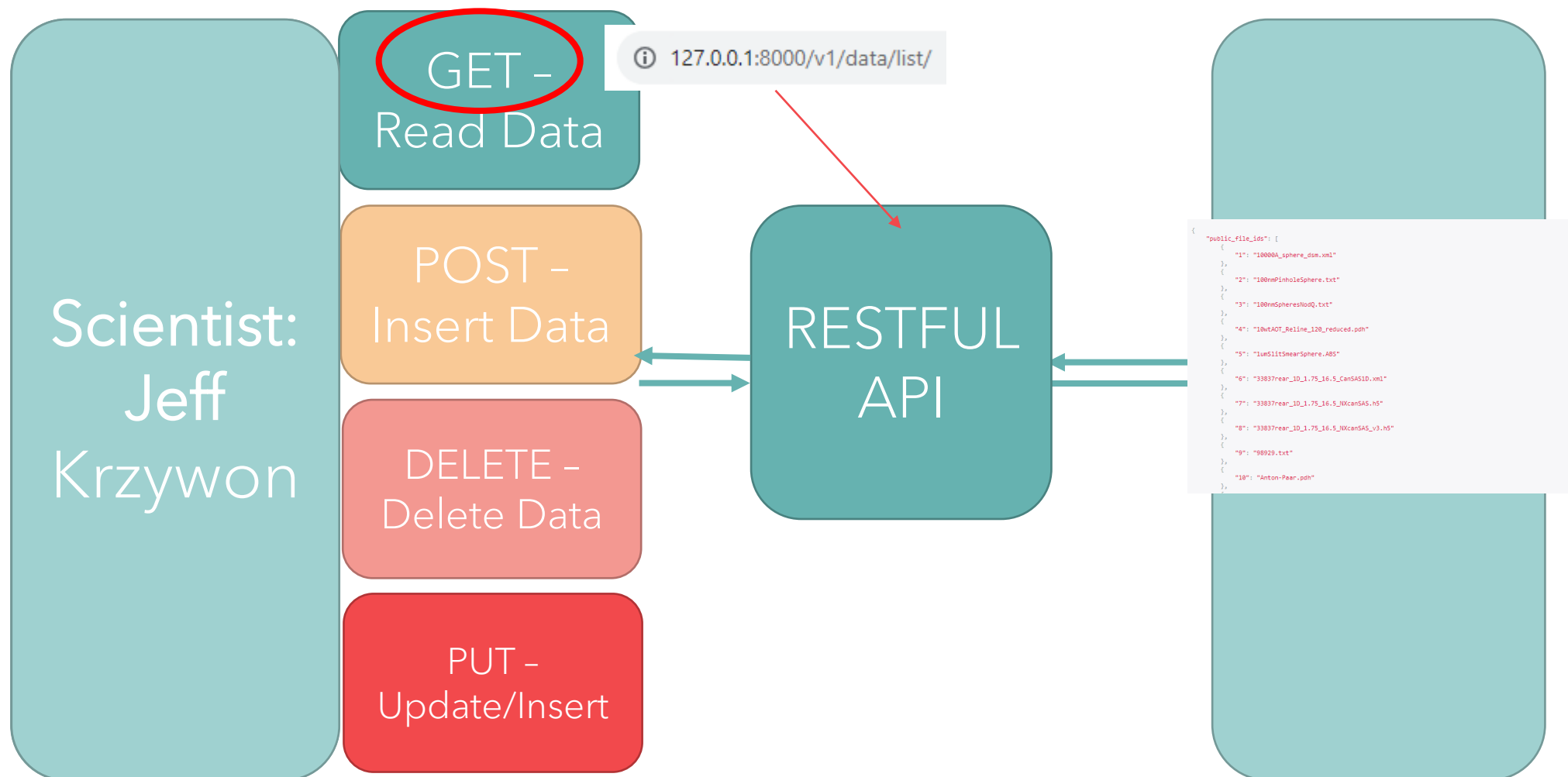
```
(base) C:\Users\tns14>python
Python 3.10.10 | packaged by Anaconda, Inc. | (main, Mar 21 2023, 18:39:17) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import requests
>>> file = open(r"C:\Users\tns14\Documents\veryrealdata.txt", 'rb')
>>> response = requests.request('POST', 'http://127.0.0.1:8000/v1/data/upload/', data = {"is_public": False}, files = {"file" : file})
>>> print(response.text)
{"current_user":null,"authenticated":false,"file_id":73,"file_alternative_name":"veryrealdata_Qq aqotB.txt","is_public":false}
>>>
```

Terminal/Command pallet

```
ests
(r"veryrealdata.txt", 'rb')
requests.request('POST', 'http://127.0.0.1:8000/v1/data/upload/',
                 data={"is_public":True}, files={'file':file})
requests.request(method='GET', url=, data=, files=)
requests.request(method='PUT', url=, data=, files=)
nse.text)
```

Script

## RESTful Framework Inputs



① 127.0.0.1:8000/v1/data/list/

```
{
  "public_file_ids": [
    {
      "1": "10000A_sphere_dsm.xml"
    },
    {
      "2": "100nmPinholeSphere.txt"
    },
    {
      "3": "100nmSpheresNodQ.txt"
    },
    {
      "22": "cyl_400_20.txt"
    },
    {
      "7": "33837rear_1D_1.75_16.5_NXcanSAS.h5"
    },
    {
      "8": "33837rear_1D_1.75_16.5_NXcanSAS_v3.h5"
    },
    {
      "9": "98929.txt"
    },
    {
      "10": "Anton-Paar.pdh"
    }
  ]
}
```

127.0.0.1:8000/v1/data/load/22/

```
{
  "cyl_400_20.txt": [
    "File: C:\\Users\\[redacted]\\sasview\\webfit\\media\\uploaded_files\\cyl_400_20.txt\nTitle: \nRun: []\nSESANS: False\nInstrum
```

False\nInstrument: \nSample:\n ID: \n Transmission: None\n Thickness: None [mm]\n Temperature: None [None]\n Position: x = None\ty = N

x = None\ty = None\tz = None [mm]\n Orientation: x = None\ty = None\tz = None [degree]\n Details:\n\nSource:\n Radiation: None\n Shape: None\n Wavel

Waveln\_max: None [nm]\n Waveln\_spread:None [percent]\n Beam\_size: x = None\ty = None\tz = None [mm]\n\n\nData:\n Type: Data1D\n X-axis:

Type: Data1D\n X-axis:  $Q$  [A<sup>-1</sup>]\n Y-axis: Intensity [cm<sup>-1</sup>]\n Length: 20\n"

127.0.0.1:8000/v1/analyze/fit/models/

```
{
  "all models": [
    "adsorbed_layer",
    "barbell",
    "bcc_paracrystal",
    "be_polyelectrolyte",
    "binary_hard_sphere",
    "broad_peak",
    "capped_cylinder",
    "core_multi_shell",
    "core_shell_bicelle",
    "core_shell_bicelle_elliptical",
    "core_shell_bicelle_elliptical_belt_rough",
    "core_shell_cylinder",
    "core_shell_ellipsoid",
    "core_shell_parallelepiped",
    "core_shell_sphere",
    "correlation_length",
    "cylinder",
    "dab",
    "ellipsoid",
    "elliptical_cylinder",
    "fcc_paracrystal",
    "flexible_cylinder",
    "flexible_cylinder_elliptical",
    "fractal",
    "fractal_core_shell",
    "fuzzy_sphere",
    "gauss_lorentz_gel",
    "gaussian_peak",
    "gel_fit",
    "guinier",
    "guinier_porod",
    "hardsphere",
    "hayter_msa",
    "hollow_cylinder",
    "hollow_rectangular_prism",
    "hollow_rectangular_prism_thin_walls",
    "lamellar",
    "lamellar_hg"
  ]
}
```

```
{
  "category" : "cylinder"
}
```

```
{
  "Cylinder models": [
    [
      "barbell",
      true
    ],
    [
      "capped_cylinder",
      true
    ],
    [
      "core_shell_bicelle",
      true
    ],
    [
      "core_shell_bicelle_elliptical",
      true
    ],
    [
      "core_shell_bicelle_elliptical_belt_rough",
      true
    ],
    [
      "core_shell_cylinder",
      true
    ],
    [
      "elliptical_cylinder",
      true
    ],
    [
      "flexible_cylinder",
      true
    ]
  ]
}
```

```
{
  "kind" : "py"
}
```

```
{
  "py models": [
    "adsorbed_layer",
    "be_polyelectrolyte",
    "broad_peak",
    "correlation_length",
    "gauss_lorentz_gel",
    "guinier_porod",
    "line",
    "peak_lorentz",
    "poly_gauss_coil",
    "polymer_excl_volume",
    "porod",
    "power_law",
    "spinodal",
    "teubner_strey",
    "two_lorentzian",
    "two_power_law",
    "unified_power_Rg"
  ]
}
```

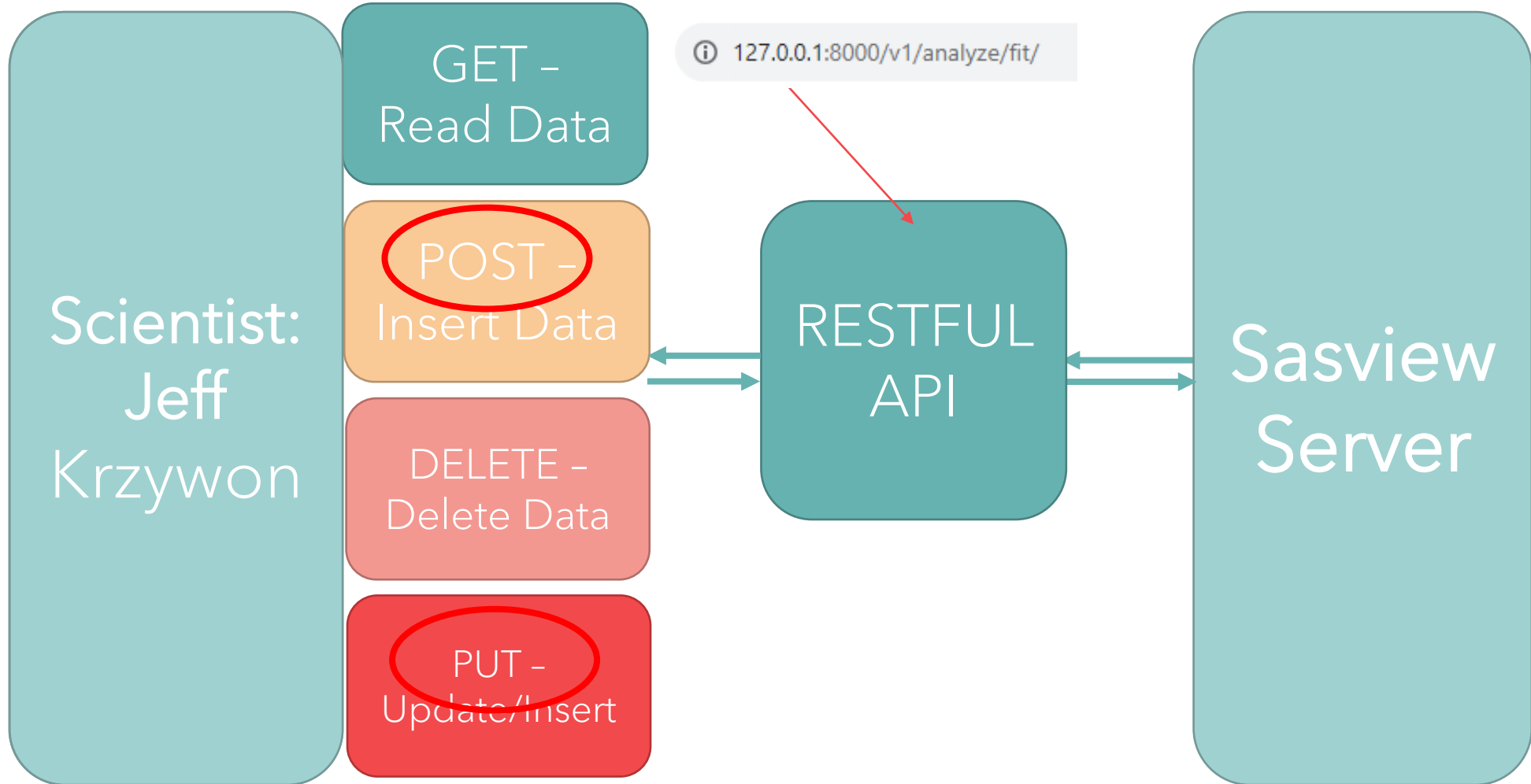


127.0.0.1:8000/v1/analyze/fit/optimizers/

```
HTTP 200 OK
Allow: GET, OPTIONS
Content-Type: application/json
Vary: Accept
```

```
{
  "optimizers": [
    [
      "amoeba",
      "de",
      "dread",
      "newton",
      "scipy.leastsq",
      "lm"
    ]
  ]
}
```

# RESTful Framework Inputs



```

{
  "model": "cylinder".
  "data_id": test_data = load_data('cyl_400_20.txt')
  "optimizer": kernel = load_model('cylinder')
  "parameter":
  [
    test_data.dy = 0.2*test_data.y
    {
      "n": pars = dict(radius=35,
      "v": length=350,
      "d": background=0.0,
      "l": scale=1.0,
      "u": sld=4.0,
      "u": sld_solvent=1.0)
    },
    {
      "n": # SET THE FITTING PARAMETERS
      "v": model.radius.range(1, 50)
      "d": model.length.range(1, 500)
      "lower_limit": 1,
      "upper_limit": 500
    }
  ],
  {
    "name": "bess ind",
    "value": 0,
    "data_type": "float",
    "lower_limit": 0,
    "upper_limit": 0,
    "value": 1.0,
    "data_type": "float",
    "value": 4.0,
    "data_type": "float",
    "value": 1.0,
    "data_type": "float",
    "value": 1.0,
    "data_type": "float"
  }
}

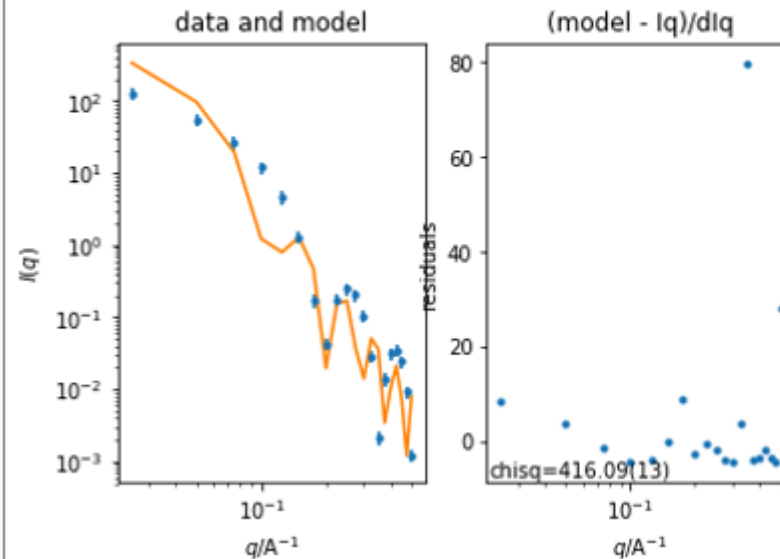
```

```
HTTP 200 OK
Allow: POST, OPTIONS
Content-Type: application/json
Vary: Accept
```

```
{
  "authenticated": false,
  "fit_id": 1,
  "results": "0.03(13)"
}
```

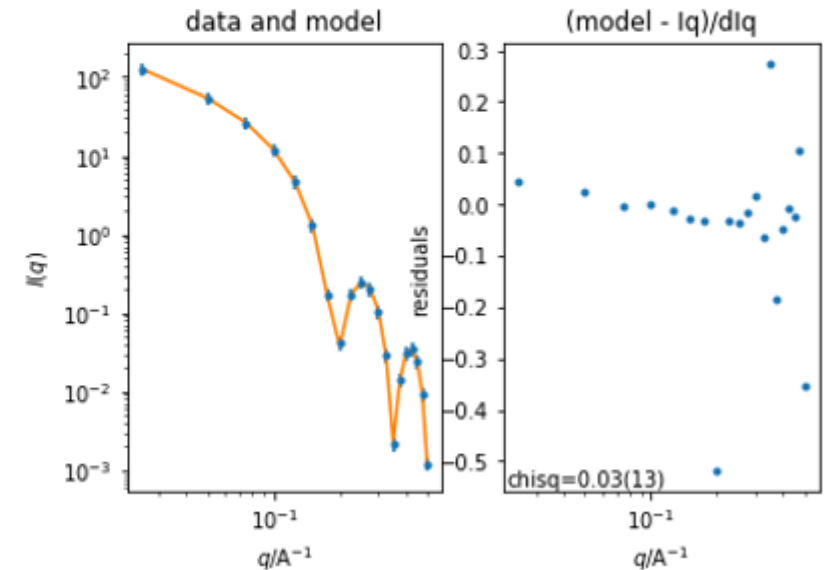
### Before fit

Initial chisq 416.09(13)



### After fit

Final chisq 0.03(13)  
length : 464.9(55)  
radius : 19.977(64)



# THANK YOU

Especially to my mentors and  
other NCNR scientists that  
helped me!



# FUTURE WORK

## Where is this living

- In Sasview repository
- Parallel to the qtGUI

## Who will pick this back up

- Any Sasview developer

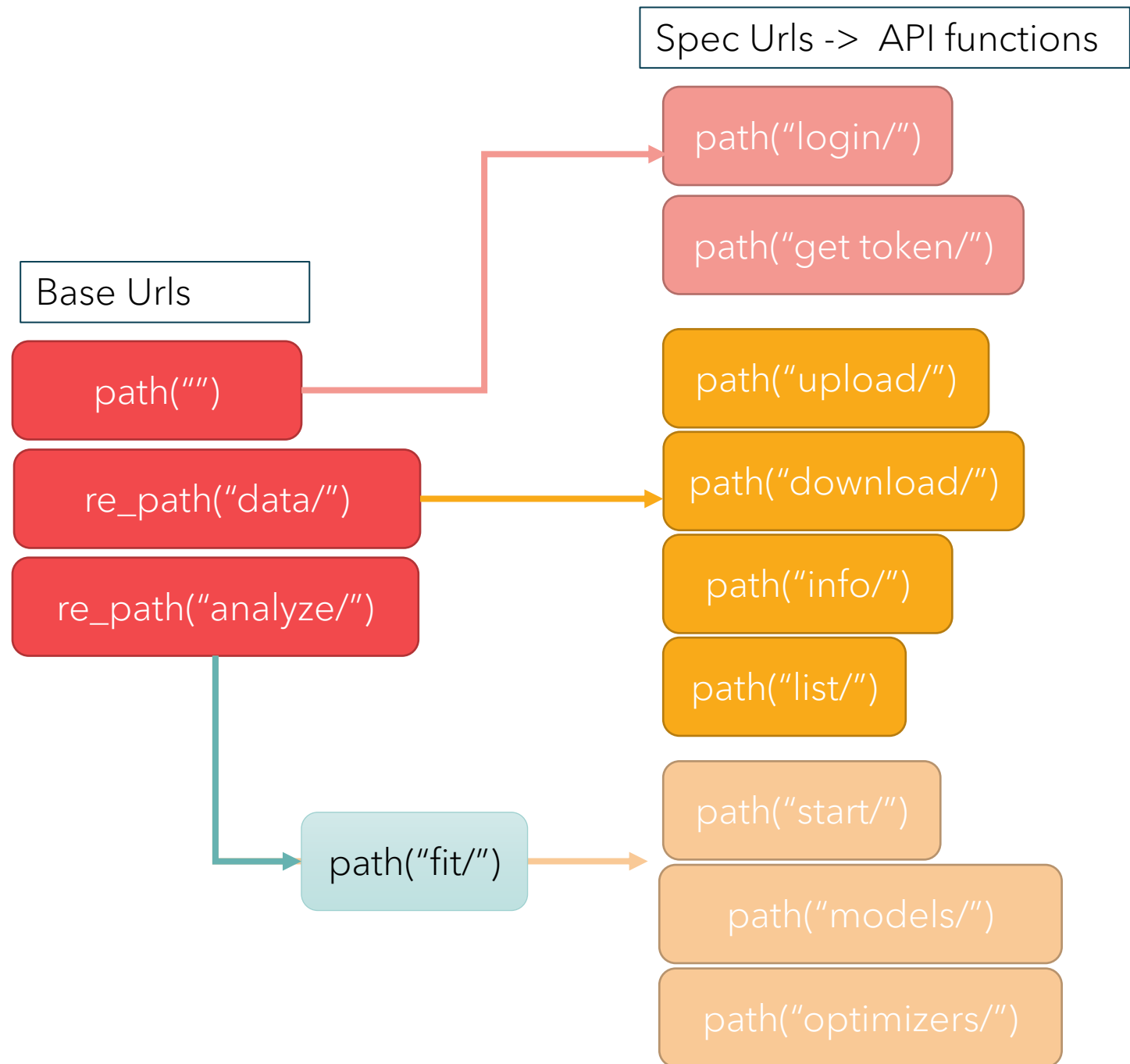
## My future work

- Working with Elizabeth Kelley to reanalyze bicelle data
- New tools

# URL PATHING

- Links everything together!
- Allows the user to get to function
- Modular
  - Add in new paths easy

Ref: [Django Rest Framework Versioning](#)



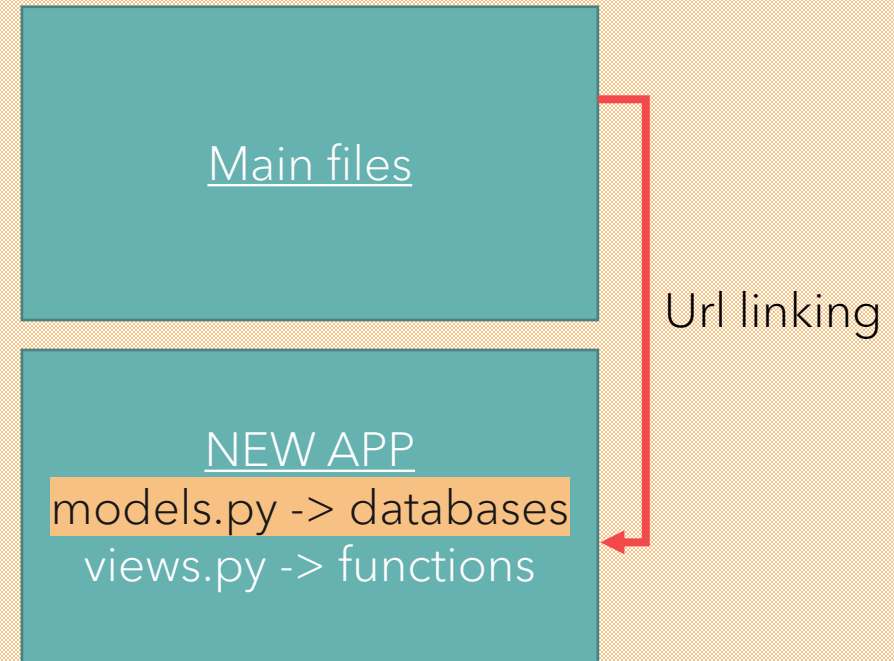
# SETTING UP DATABASES

## Models for Django

- Define what the user inputs
- Make sure user can't put in malicious data

Data

Data id	user	file	file_name	is_public
1	None	<File_obj>	Data.txt	True
2	None			
3...	None			



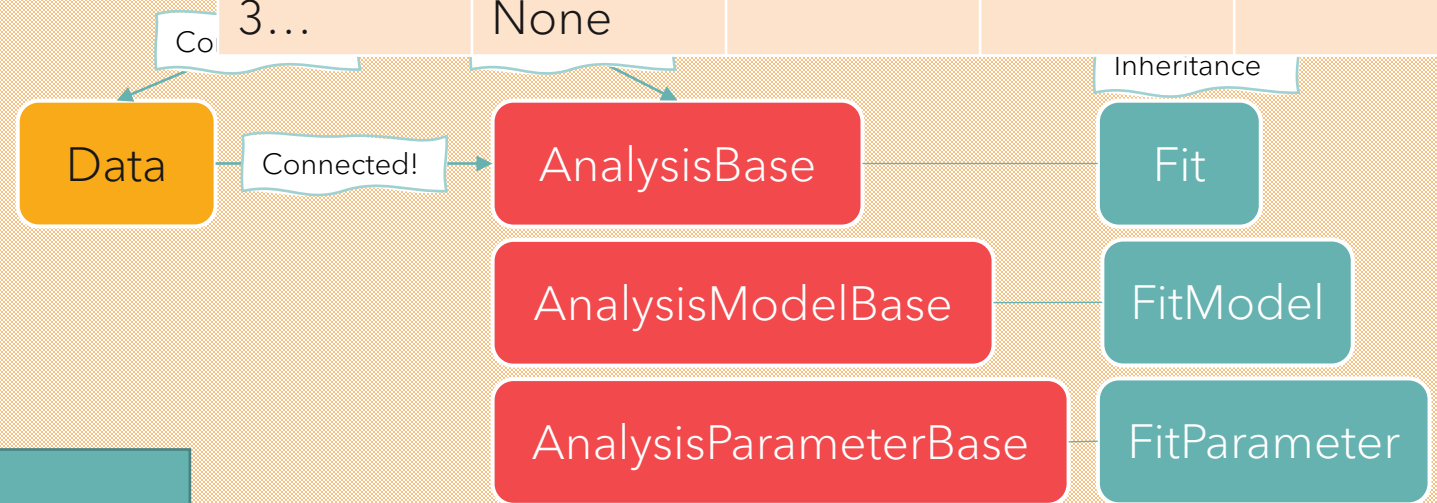


# SETTING UP DATABASES

## Models for Django

- Data - holds imported data for users
- Analysis - holds data for analysis
- Fit - extends analysis, specific data for fit

Data id	user	file	file_name	is_public
1	None	<File_obj>	Data.txt	True
2	None			
3...	None			



NEW APP  
models.py -> databases  
views.py -> functions

# VIEWS

- Views hold the meat of the code:
- Takes user request <- holds data
- Returns response <- specified by function

NEW APP  
models.py -> databases  
views.py -> functions

🌐 127.0.0.1:8000/v1/data/upload/?data\_id=1

```
@api_view(['POST', 'PUT'])
def upload(request, data_id=None, version=None):
    #saves file
    if request.method == 'POST':
        form = DataForm(request.data, request.FILES)
        if form.is_valid():
            form.save()
        db = Data.objects.get(pk = form.instance.pk)
        if request.user.is_authenticated:
            serializer = DataSerializer(db, data=
                {"file_name":os.path.basename(form.instance
                |"current_user" : request.user.id})
        else:
            serializer = DataSerializer(db, data=
                {"file_name":os.path.basename(form.instance
```

# VIEWS

- Views hold the meat of the code:
- Takes user request <- holds data
- Returns response <- specified by function

NEW APP  
models.py -> databases  
views.py -> functions

🌐 127.0.0.1:8000/v1/data/upload/?data\_id=1

```
@api_view(['POST', 'PUT'])  
def upload(request, data_id = None, version = None):
```

request.user

request.data

request.FILES

"id" : 1  
"username" : "Harry"  
"password" : "123?"  
"is\_logged\_in" : True

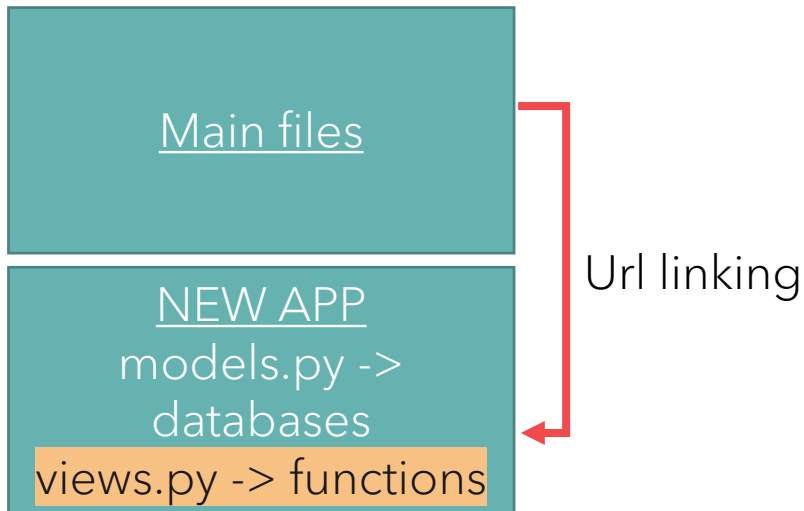
"is\_public" : False  
"file\_name" : "hello.txt"

"file" : file object

Data

# VIEWS

- Translator
- Puts into databases
- Defines how user's data looks



🌐 127.0.0.1:8000/v1/data/upload/?data\_id=1

```
@api_view(['POST', 'PUT'])  
def upload(request, data_id = None, version = None):
```

request.user

request.data

request.FILES

"id" : 1  
"username" : "Harry"  
"password" : "123?"  
"is\_logged\_in" : True

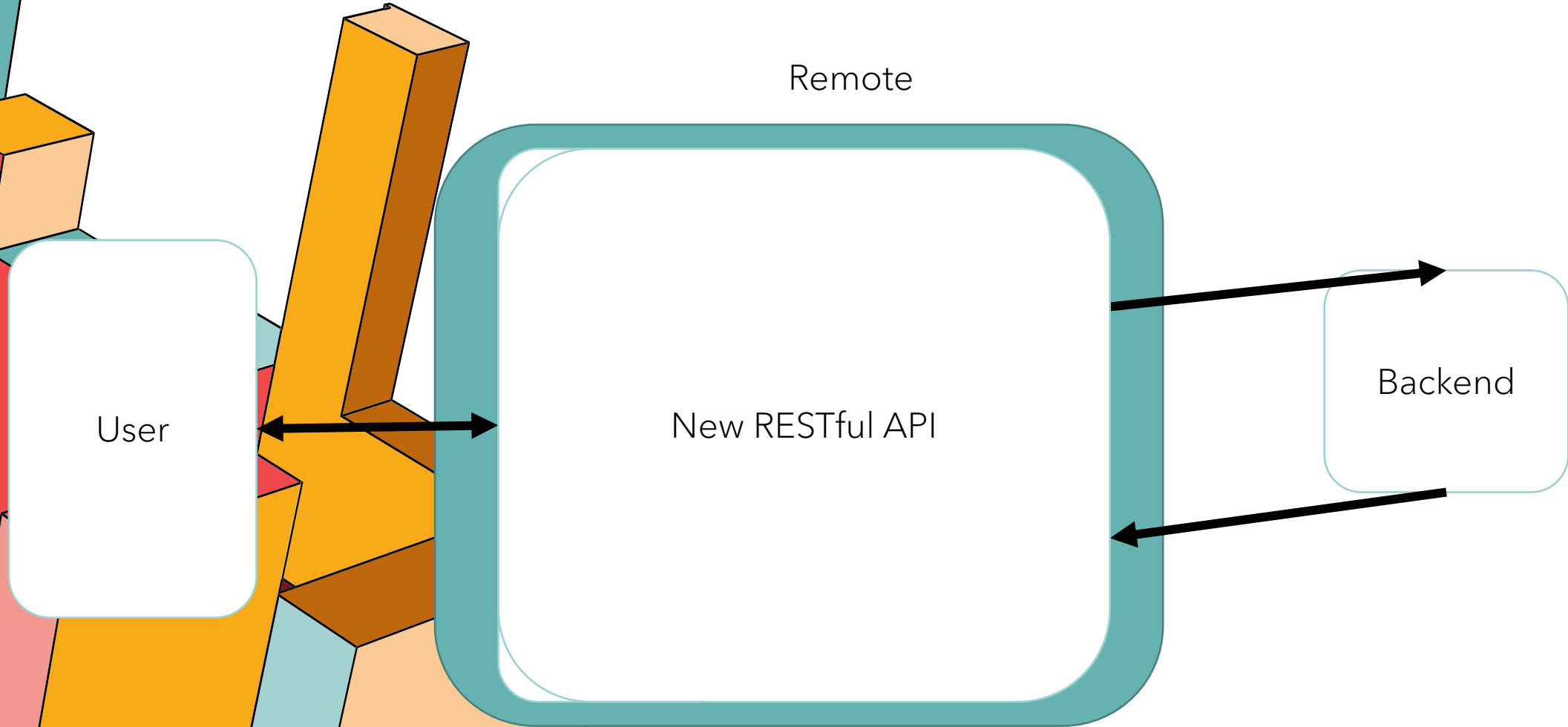
"is\_public" : False  
"file\_name" : "hello.txt"

"file" : file object

Data

# Proposed Layout

Remote



# UPLOADING DATA

**Example  
Data**

->

```
def parse_1D():  
    dir_1d = os.path.join(EXAMPLE_DATA_DIR, "example_data", "1d_data")  
    for file_path in glob(os.path.join(dir_1d)):  
        upload_file(file_path)
```

```
example_data  
  L 1d_data  
    L file1  
    L file2
```

```
def upload_file(file_path):  
    data_file = Data.objects.create(file_name = file_name, is_public = True)  
    data_file.file.save(file_name, open(file_path, 'rb'))
```

Data



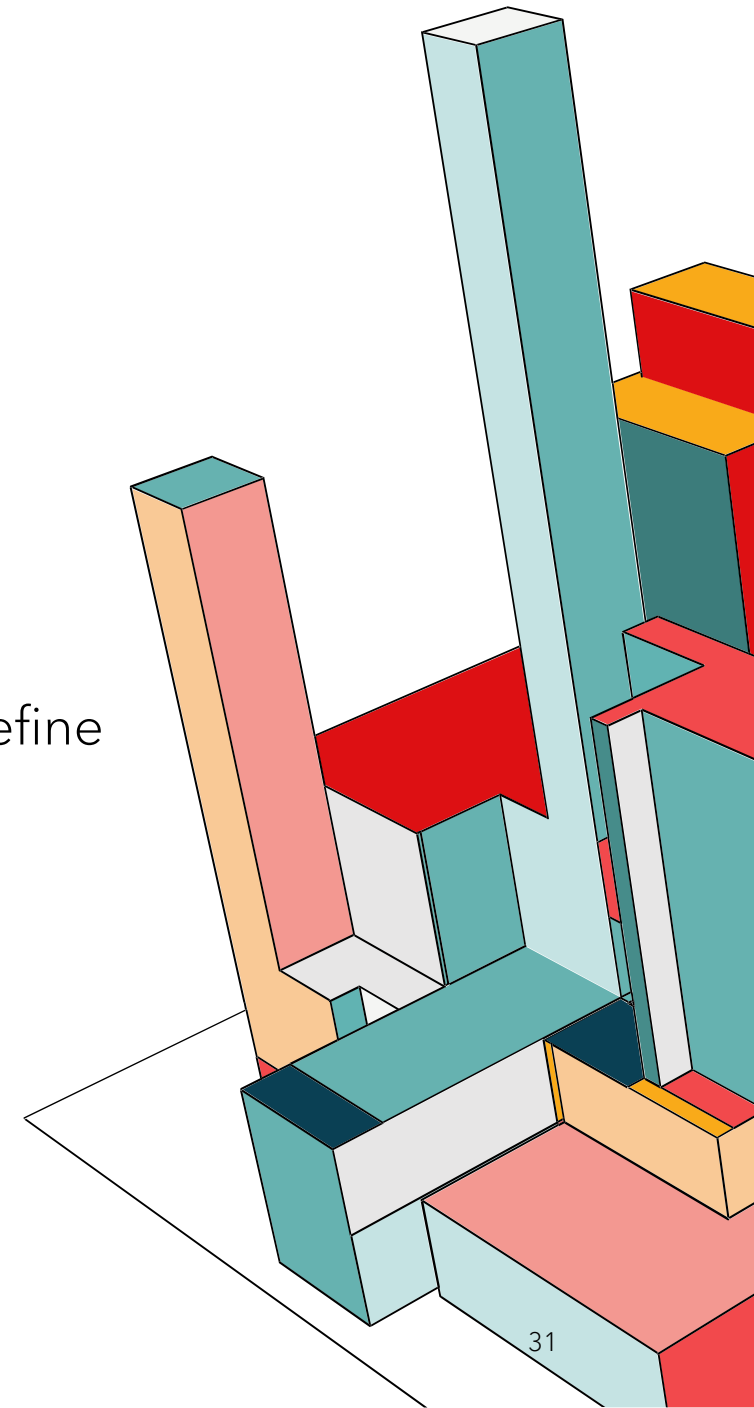
# OTHER FACTORS

## Why Django?

- Web framework
- Batteries Included
- DRY (Don't Repeat Yourself)
- Database backend
- Future View integration

## International

- Users all over the world!
- Different wants
  - Need something to define "public"
- Save Data and Fits



# OTHER FACTORS

## What web framework?

- Django
- Flask
- What's the best?



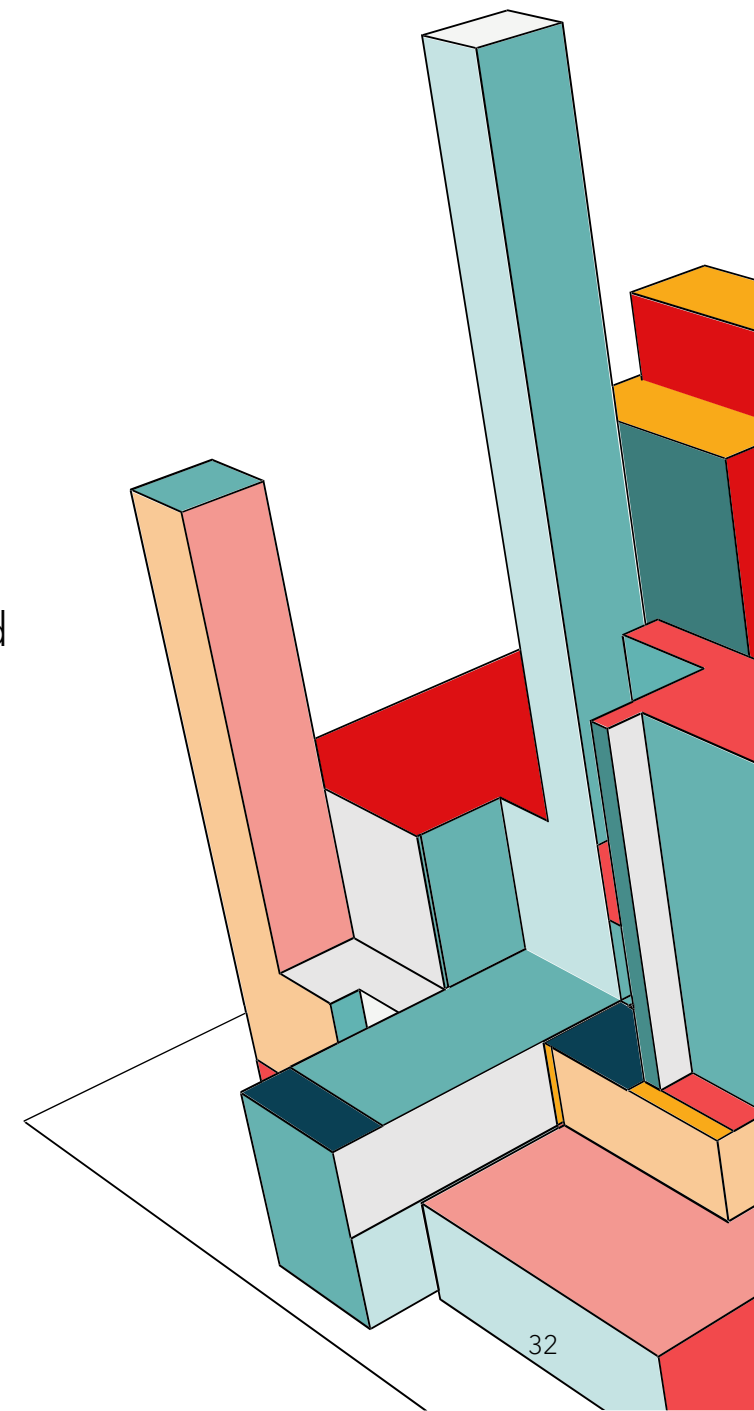
## International

- Users all over the world!
- Different wants
  - Need something to define "public"



## Why Django?

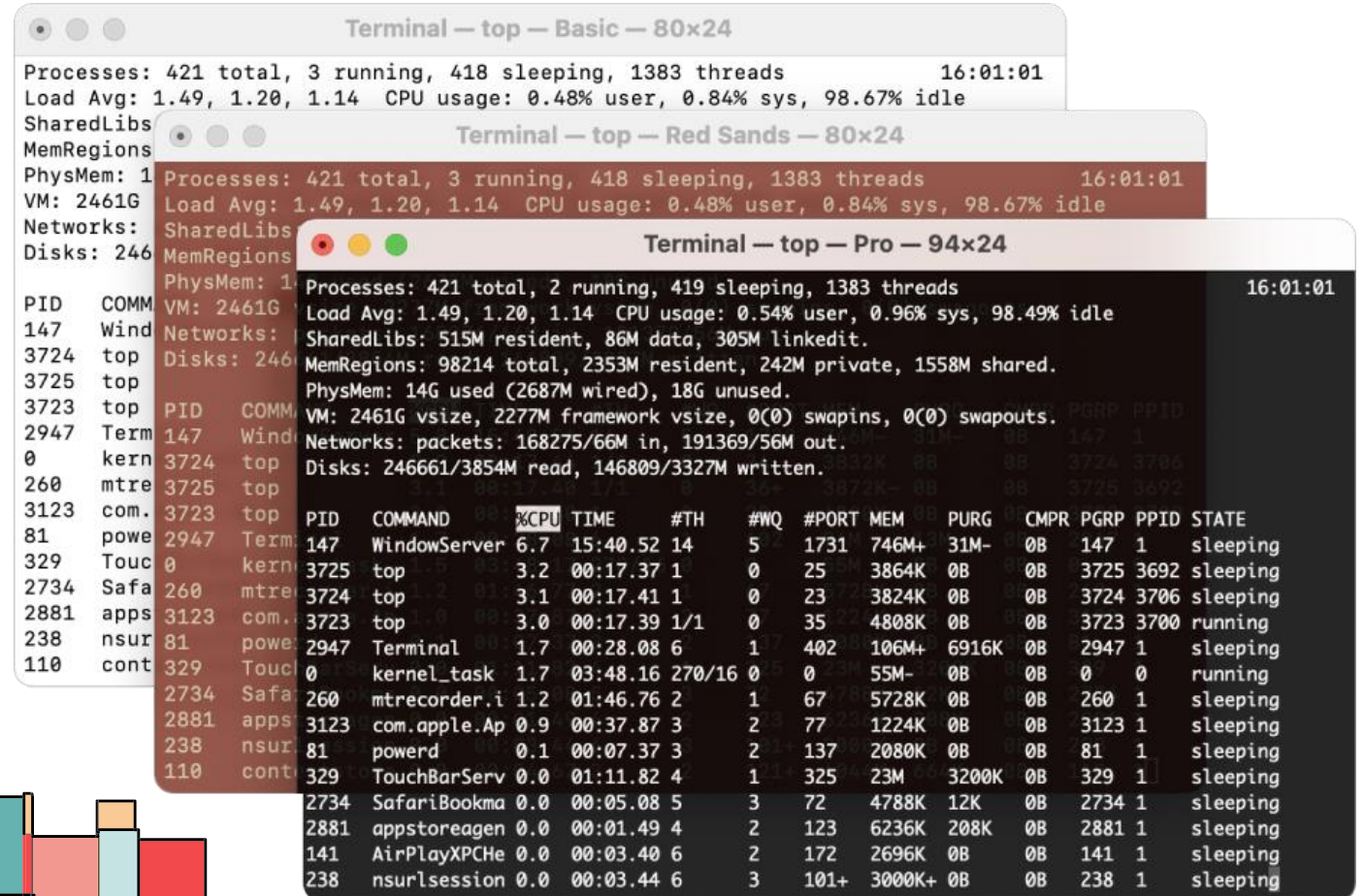
- Defined database backend
- Allows for the future development of web application rather than just interface





# BACKEND?

terminal



The image shows three overlapping terminal windows. The top window, titled 'Terminal — top — Basic — 80x24', displays system statistics: 'Processes: 421 total, 3 running, 418 sleeping, 1383 threads' and 'Load Avg: 1.49, 1.20, 1.14 CPU usage: 0.48% user, 0.84% sys, 98.67% idle'. The middle window, 'Terminal — top — Red Sands — 80x24', shows identical statistics. The bottom window, 'Terminal — top — Pro — 94x24', shows a detailed process list with columns for PID, COMMAND, %CPU, TIME, #TH, #WQ, #PORT, MEM, PURG, CMPR, PGRP, PPID, and STATE. The process list includes 'WindowServer', 'kernel\_task', 'mtrecorder.i', 'com.apple.Ap', 'powerd', 'TouchBarServ', 'SafariBookma', 'appstoreagen', 'AirPlayXPCh', and 'nsurlsession'.

```
Processes: 421 total, 3 running, 418 sleeping, 1383 threads      16:01:01
Load Avg: 1.49, 1.20, 1.14  CPU usage: 0.48% user, 0.84% sys, 98.67% idle
SharedLibs
MemRegions
PhysMem: 1
VM: 2461G
Networks:
Disks: 246

PID  COMM
147  Wind
3724 top
3725 top
3723 top
2947 Term
0    kern
260  mtre
3123 com.
81   powe
329  Touc
2734 Safa
2881 apps
238  nsur
110  cont

PID  COMMAND  %CPU  TIME    #TH   #WQ   #PORT  MEM    PURG   CMPR  PGRP  PPID  STATE
147  WindowServer 6.7   15:40.52 14    5     1731  746M+  31M-  0B   147  1    sleeping
3725 top          3.2   00:17.37 1      0     25    3864K  0B    0B   3725 3692 sleeping
3724 top          3.1   00:17.41 1      0     23    3824K  0B    0B   3724 3706 sleeping
3723 top          3.0   00:17.39 1/1    0     35    4808K  0B    0B   3723 3700 running
2947 Terminal    1.7   00:28.08 6      1     402   106M+  6916K  0B   2947 1    sleeping
0    kernel_task  1.7   03:48.16 270/16 0      0     55M-  0B    0B   0    0    running
260  mtrecorder.i 1.2   01:46.76 2      1     67    5728K  0B    0B   260  1    sleeping
3123 com.apple.Ap 0.9   00:37.87 3      2     77    1224K  0B    0B   3123 1    sleeping
81   powerd       0.1   00:07.37 3      2     137   2080K  0B    0B   81   1    sleeping
329  TouchBarServ 0.0   01:11.82 4      1     325   23M    3200K  0B   329  1    sleeping
2734 SafariBookma 0.0   00:05.08 5      3     72    4788K  12K   0B   2734 1    sleeping
2881 appstoreagen 0.0   00:01.49 4      2     123   6236K  208K  0B   2881 1    sleeping
141  AirPlayXPCh  0.0   00:03.40 6      2     172   2696K  0B    0B   141  1    sleeping
238  nsurlsession 0.0   00:03.44 6      3     101+  3000K+ 0B    0B   238  1    sleeping
```