Lab 1 Report

Title: Comparison of three different API
Notice: Dr. Bryan Runck
Author: Liang-Ting Chen
Date: Feb 10, 2021

Project Repository:

https://github.com/chen6761/GIS5572

Abstract

In the lab, I compare and contrast the conceptual models from three different APIs. There are Minnesota Geospatial Commons, Google Places, and North Dakota Agricultural Weather Network (NDAWN). Then, I write a code in Jupyter Notebooks for each API to catch their data programmatically.

Problem Statement

In the lab, I focus on the models for the different APIs and the way to get their data through the programming.

First, I try to draw the conceptual models for each API to find the difference between them. Then, I analyze the website with the conceptual models I did to find their organization. After that, I write the code for each of them in the Jupyter Notebooks to get the data by programming.

Input Data

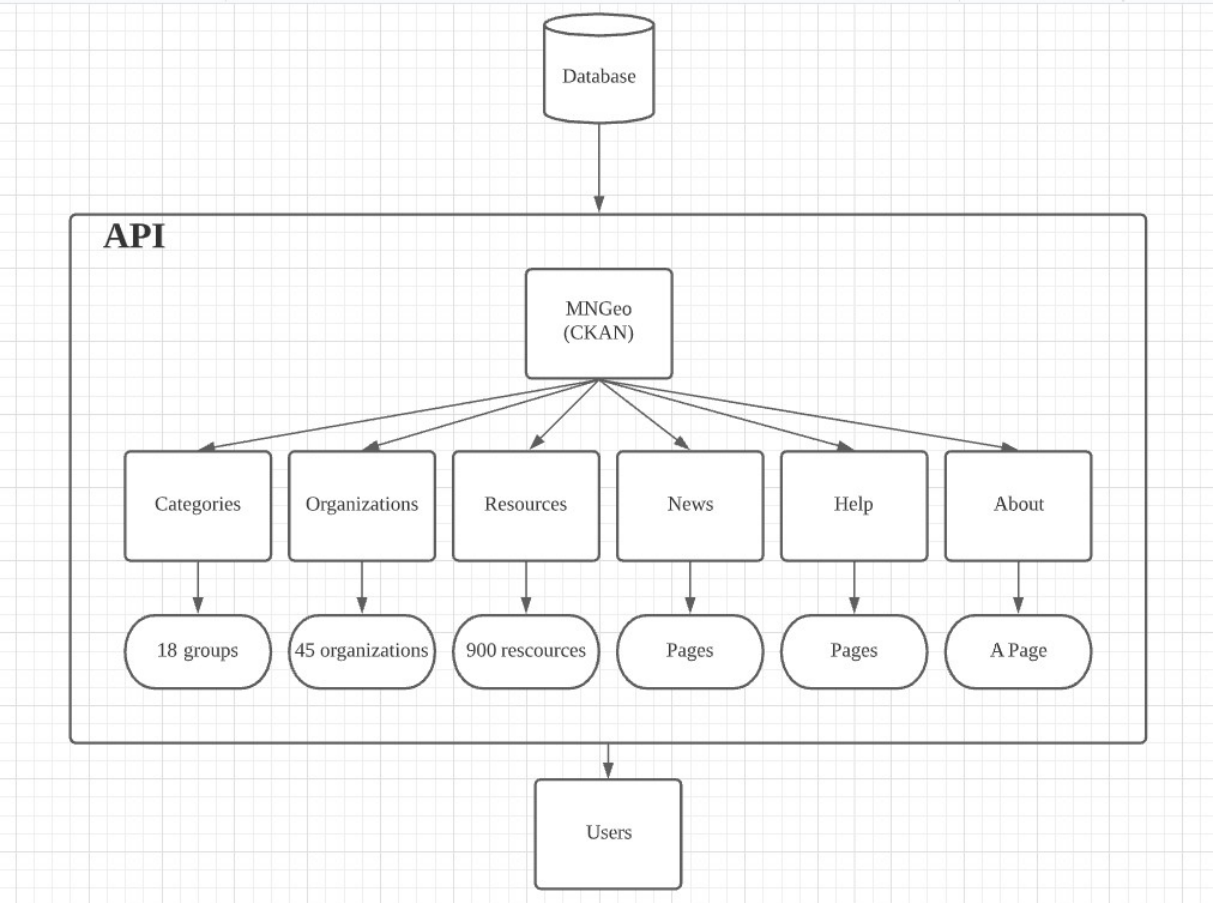
In the lab, I follow the instructor to use Minnesota Geospatial Commons, Google Places, and NDAWN as the targets to do my analysis. The table below shows the link of them.

Table. The input data

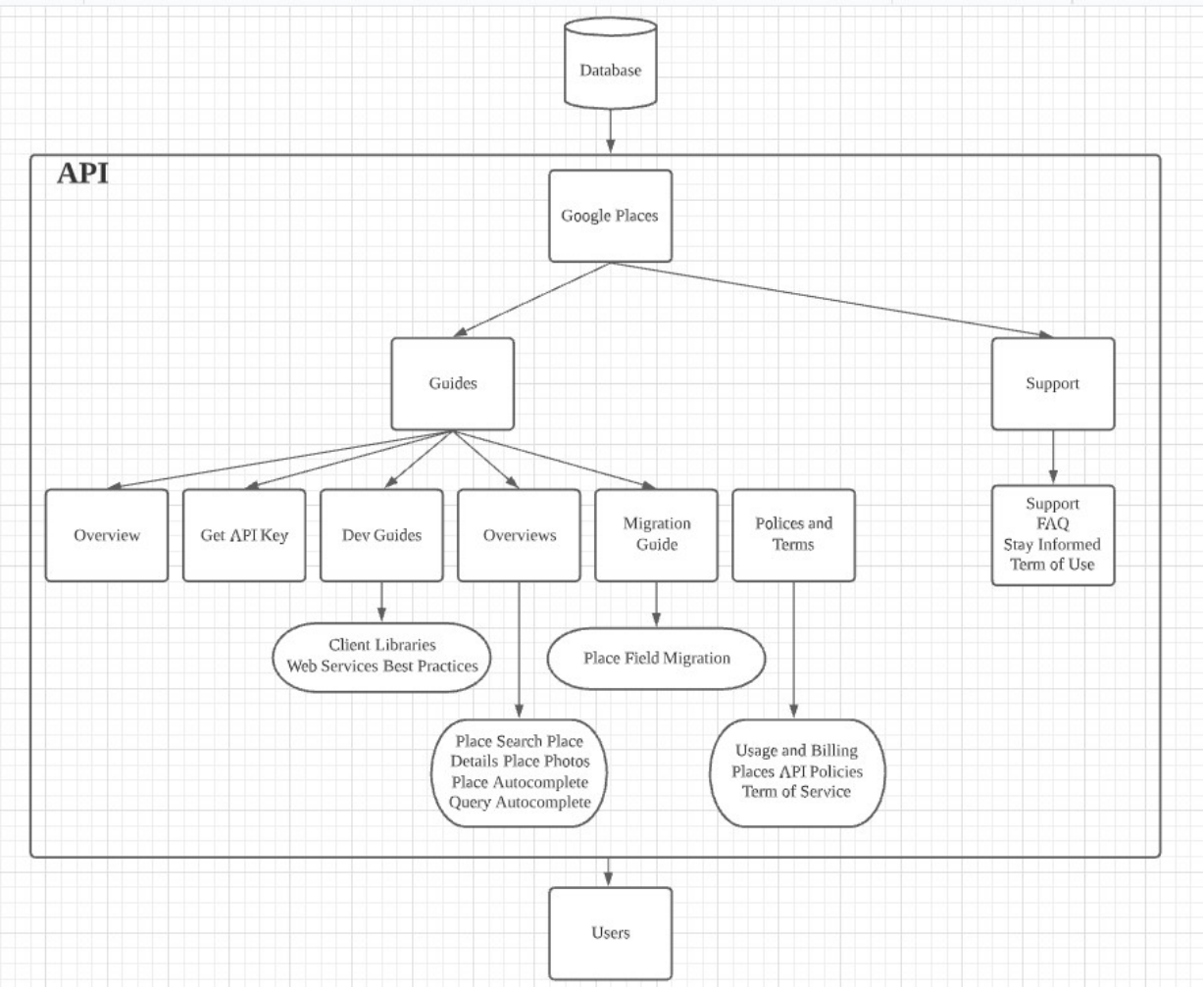
#	Title	Purpose in Analysis	Link to Source
1	Minnesota Geospatial Commons	Make the conceptual models and catch their data programmatically	MNGeo
2	Google Places		Google Places
3	NDAWN		NDAWN

Methods

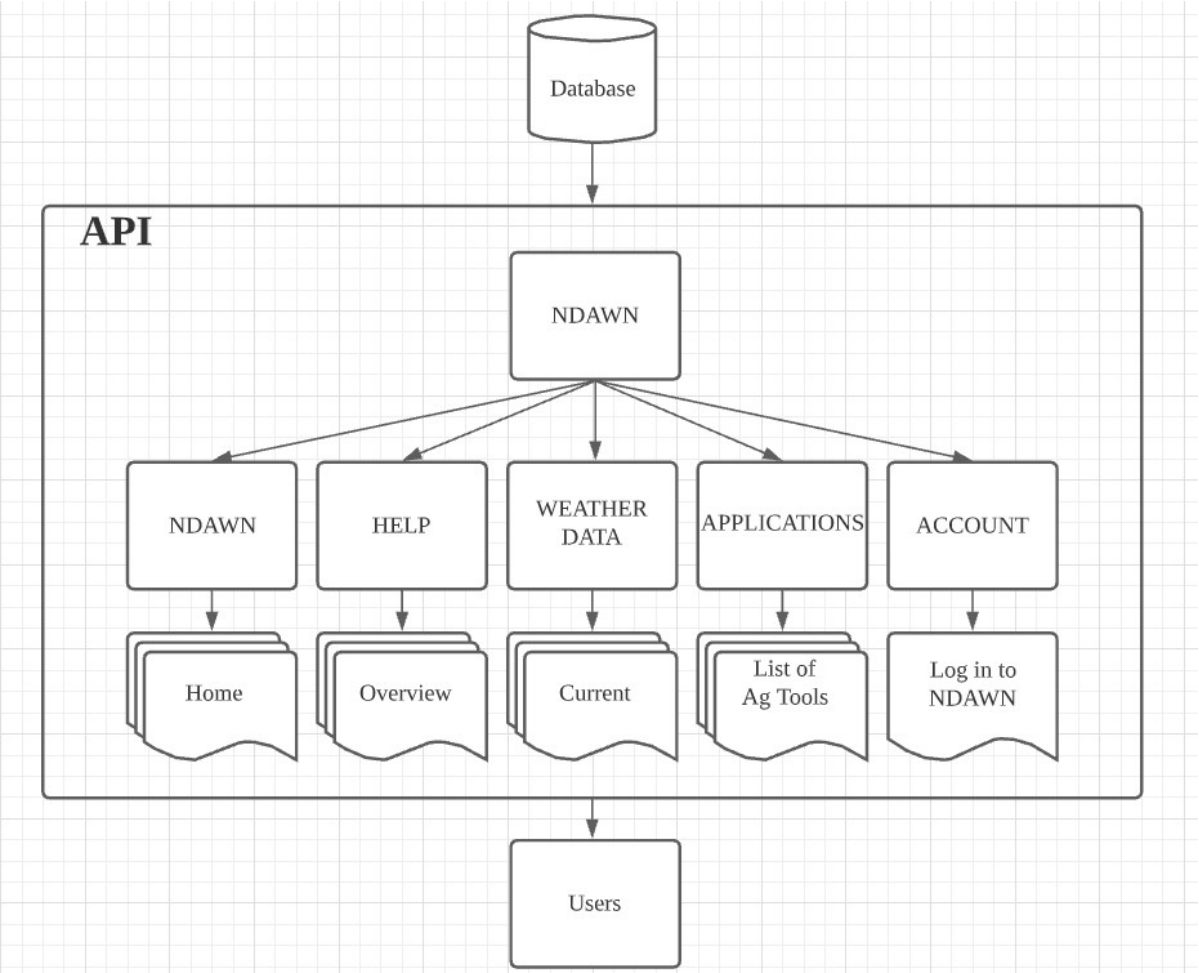
Minnesota Geospatial Commons



Google Places



NDAWN


Database
API
NDAWN
NDAWN
HELP
WEATHER DATA
APPLICATIONS
ACCOUNT
Home
Overview
Current
List of Ag Tools
Log in to NDAWN
Users

Results

	Document	Account	Export Table	Download
MNGeo	V			V
Google Places	V	V		V
NDAWN	V	V	V	V

All of the APIs are easy to read, and users can understand their functions clearly because they have documents to introduce their usage. Besides, users can download the data they need via these APIs, something special is that Google Places provides API Keys for users to apply in their work; MNGeo and NDAWN provide more intuitive data for users. The difference between these API is that one of them can export the .csv files (NDAWN), some of them include the account certification (NDAWN and Google Places), and one of them build its API via CKAN that users can catch data from it based on their code. (MNGeo)

Results Verification

Take the conceptual models above as a reference, I write the codes to get the data from each API programmatically. In this part, I learned how to use the parsing tools and understand the relationship between the API and codes.

NDAWN

In this API, the searching result shows on in the URL, like the "station=65&begin_date=2020-01&count=24&quick_pick=&ttype=monthly", and I can change some intuitive variable to get the data I want. Then, I use beautifulsoup to analyze the script code behind the API and find the URL for the "Export CSV File". (Wang, 2018) After combining the home page URL and the downloading URL, I can save the searching result as a CSV file to my local computer.

Get data from NDWAN

In [1]:
```
# Import the necessary modules
import requests
from bs4 import BeautifulSoup
```

In [2]:
```
# Set the variable for the URL of data page
data = "https://ndawn.ndsu.nodak.edu/get-table.html?station=65&begin_date=2020-01&count=24&quick_pick=&ttype=monthly&variable=md
```

In [3]:
```
# Get the website data
page = requests.get(data)
```

In [4]:
```
# Build the BeautifulSoup item from the website data
parser = BeautifulSoup(page.text,"html.parser")
```

In [5]:
```
# Find the string includes "Export CSV File"
target = parser.find_all("a", string="Export CSV File")
```

In [6]:
```
# Add the original url in front of the link
for link in target:
    url="https://ndawn.ndsu.nodak.edu"+link.get('href')
```

In [7]:
```
# Get the data and save it as a CSV file
r = requests.get(url, allow_redirects=True)
open('weather.csv', 'wb').write(r.content)
```

Out[7]: 3334

MNGeo

I change the way to read the website data. In the NDAWN, I used requests.get(data), however, I meet some SSL problem and decide to change a method to get the website data. After googling the question, I find I can use the urllib module to get the data as well. (Bodnar, 2020) This time I find the shapefile downloading link is under a class, and I use the same way to save it as a zip file.

Get data from MNGeo

In [1]:
```
# Import the necessary modules
from urllib.request import urlopen
from bs4 import BeautifulSoup
import requests
```

In [2]:
```
# Use urlopen to get the data and save it as a variable
response = urlopen("https://gisdata.mn.gov/dataset/env-land-application-sites")
html = response.read()
```

In [3]:
```
# Build the BeautifulSoup item from the website data
parser = BeautifulSoup(html.decode("utf-8"), "html.parser")
```

In [4]:
```
# Find the class for downloading the shapefile
target = parser.find_all('a', class_='SHP btn btn-primary dropdown-toggle resource-url-analytics')
```

In [5]:
```
# Get the downloading link from the class above
for link in target:
    url=link.get('href')
```

In [6]:
```
# Get the data and save it as a zip file
r = requests.get(url, allow_redirects=True)
open('shapefile1.zip', 'wb').write(r.content)
```

Out[6]: 2313322

Google Place

In this API, I use its text search tool and find the results URL includes my searching words. After adding my API Key behind the URL, I can get a script code page. The page includes the information I search but it not easy to read, so I use beautifulsoup to deal with it and catch the necessary data. In the lab, I search the Japanese restaurant near me and get their name and address from the API to save it as a CSV file.

Get data from Google Places

In [1]:
```
# Import the necessary modules
import requests
from bs4 import BeautifulSoup
import pandas as pd
```

In [2]:
```
# Set the variable for the URL of data page
data = "https://maps.googleapis.com/maps/api/place/textsearch/xml?query=American+restaurant&key=AIzaSyAVJ57_319KeYnOFo4L3Elx15Tb
```

In [3]:
```
# Get the website data
page = requests.get(data)
```

In [4]:
```
# Build the BeautifulSoup item from the website data
sp = BeautifulSoup(page.text,"html.parser")
```

In [5]:
```
# Find the target tag for name
name = sp.find_all("name")
```

In [6]:
```
# Create a empty list to catch the name
output_name = []
for name_col in name:
    output_name.append(name_col.text)
```

In [7]:
```
# Find the target tag for address
addr = sp.find_all("formatted_address")
```

In [8]:
```
# Create a empty list to catch the address
output_addr = []
for addr_col in addr:
    output_addr.append(addr_col.text)
```

In [9]:
```
# Combine two list as a datafrom and save it as a CSV file
output = {'Name': output_name, 'Address': output_addr}
df = pd.DataFrame(output)
df.to_csv('restaurant1.csv')
```

Discussion and Conclusion

After doing this lab, I become more familiar with the model of API and how to analyze its source code. The practice of getting the data programmatically is useful for me. I think it is an efficient way to get lots of data, moreover, dynamically to refresh the downloading data. It can be used in

the project that needs to show some in time data, like weather, traffic, and wildfire. I hope I can use the parsing more familiar and apply it in my project in the future.

References

Bodnar, J. (2020, July 6). Python urllib3 tutorial - accessing web resources via HTTP. ZetCode. https://zetcode.com/python/urllib3/

Wang, G. T. (2018, February 4). Python 使用 Beautiful Soup 抓取與解析網頁資料，開發網路爬蟲教學 - 頁 2，共 3. G. T. Wang. https://blog.gtwang.org/programming/python-beautiful-soup-module-scrape-web-pages-tutorial/2/

Self-score

Category	Description	Points Possible	Score
Structural Elements	All elements of a lab report are included **(2 points each)**: Title, Notice: Dr. Bryan Runck, Author, Project Repository, Date, Abstract, Problem Statement, Input Data w/ tables, Methods w/ Data, Flow Diagrams, Results, Results Verification, Discussion and Conclusion, References in common format, Self-score	28	28
Clarity of Content	Each element above is executed at a professional level so that someone can understand the goal, data, methods, results, and their validity and implications in a 5 minute reading at a cursory-level, and in a 30 minute meeting at a deep level **(12 points)**. There is a clear connection from data to results to discussion and conclusion **(12 points)**.	24	23
Reproducibility	Results are completely reproducible by someone with basic GIS training. There is no ambiguity in data flow or rationale for data operations. Every step is documented and justified.	28	27
Verification	Results are correct in that they have been verified in comparison to some standard. The standard is clearly stated **(10 points)**, the method of comparison is clearly stated **(5 points)**, and the result of verification is clearly stated **(5 points)**.	20	19
		100	97