Magnetic Polarity Inversion Lines API

Backend Setup

We used Django with the REST API Framework for the backend services.
As an IDE, we used Visual Studio Code. All of the below commands were completed on IDE Terminal.

Create a directory for the project.

```
cd   mpil_project
```

Build a virtual environment in that directory

We create a virtual environment because having only the packages we need eliminates all chances for us to experience unpleasant global installation and package collision errors and focus on code.

```
python -m venv env
```

Install Django

```
pip install django
```

Activate the virtual environment

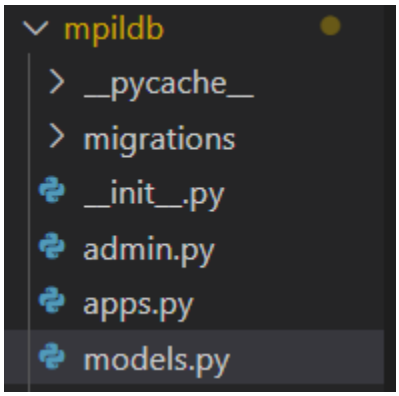
```
env/Scripts/activate
```

Start our Django project

```
django-admin startproject mpil_api
```

Move to the project directory and start our Django app

```
python manage.py startapp mpildb
```



This folder is for our database configuration since we are developing a code-first architecture. Here we will describe our models (tables in a dbms language).

Database Information

For this system, PostgreSQL was used as a database.

As a DBMS, we used pgAdmin 4.

Coding for the database creation

Python was used on the DMLab server to access the dataset.

We created separate tables for each year of occurrence. To get the necessary years based on the existing dataset, we use this Python code.

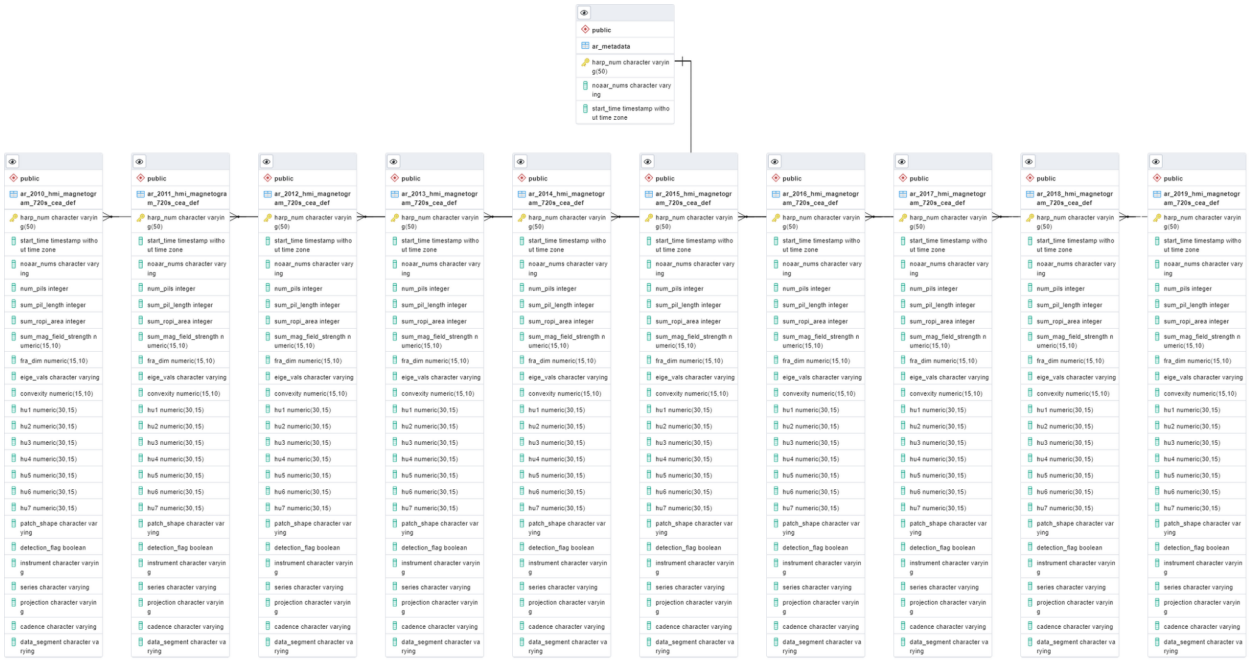
- Change the folder path if different.
- Note that if the filenames are changed, you have to modify the filename parts of the code below.

```
import os
import pandas as pd
rootdir = '/data/MPIL/HMI_PIL/'
start_with = 'AR_'
ends_with = '_HMI_magnetogram_720s_CEA_def'
counter = 0
years_all = pd.Series()

for subdir, dirs, files in os.walk(rootdir):
    for file in files:
        if(not(file.endswith('not_proceed_HARP.csv') or
               file.endswith('HARP_length.csv')) and file.endswith('.csv')):
            region = subdir.split('HMI_PIL/')[-1]
            df = pd.read_csv(os.path.join(subdir, file), index_col = None,
                             header = 0,
                             sep = ',')
            years =  pd.to_datetime(df.iloc[:, 0]).dt.year
            years_all = years_all.append(pd.Series(years.unique()))
            print(years_all.unique())
```

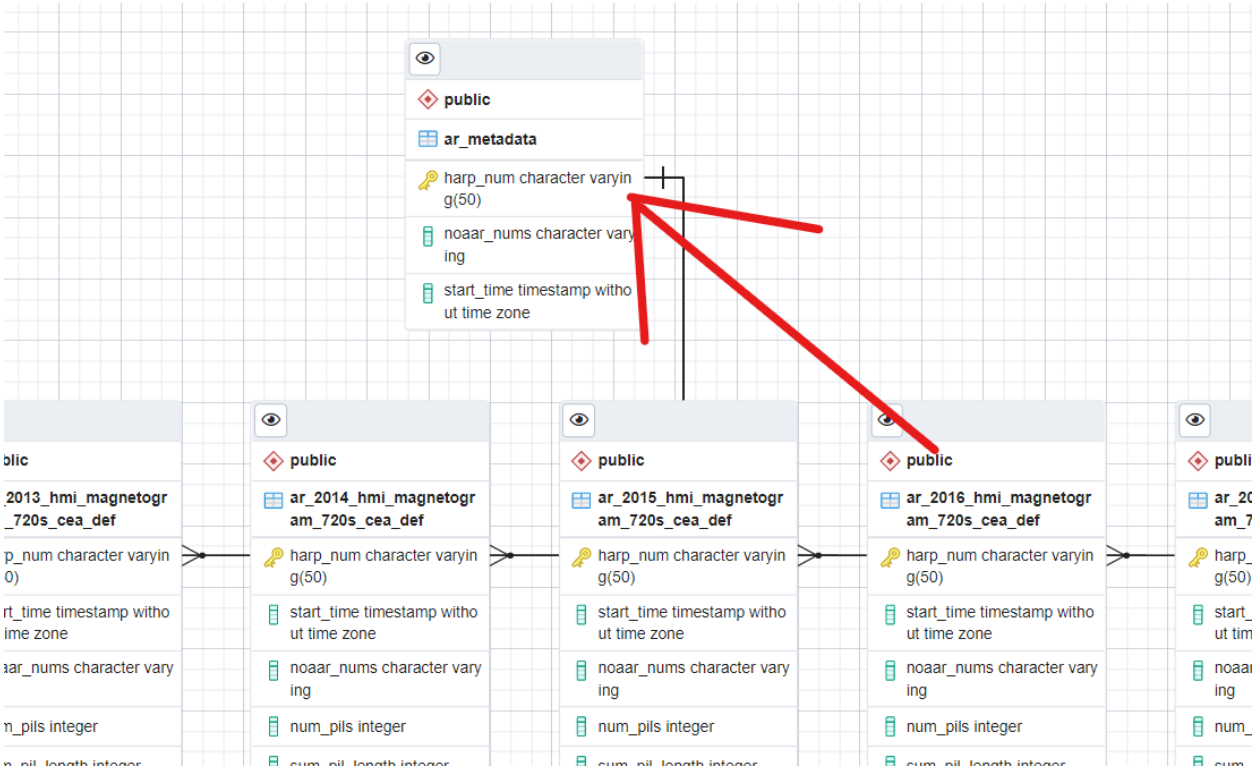
Output example: [2010 2011 2012 2013 2014 2015 2016 2017 2018 2019]

Database Scheme



Let's zoom in

And we can see the One-To-Many relations between our yearly and metadata tables



Tables architecture

DISCLAIMER: THE DATA SAMPLE BELOW IS RANDOMLY GENERATED. ONLY FOR DEMONSTRATION PURPOSES.

armetada

harp_num	noaar_nums	start_time
377	1	2/11/2011 2:24