CoolAir API

This API builds on the Airline delays dataset made available by the CORGIS dataset project, itself a curated version of the data published by the Bureau of Transportation Statistics of the US Government.

This project consists of two parts, the backend code and the frontend code. The backend was created using Python and Django. Django is a high-level Web framework that has scalability, security and speed as its main concerns. Django includes dozens of extras you can use to handle common Web development tasks. It also takes care of user authentication, content administration, site maps, RSS feeds, and many more tasks.

Installation

1) you must have installed: python 3.7.3, Django 2.1.7
2) git clone https://github.com/shoubh/CoolAirBackend.git
3) open windows power shell in project containing folder
4) activate Django environment: ./Scripts/activate
5) cd src/coolair
6) python manage.py runserver

Data

The dataset which was used to create this project was the Airlines JSON Library from the CORGIS Dataset Project. This project shows information about U.S flights. More specifically there is information regarding:

- The number of delays and cancellations caused by a previous flight with the same aircraft arriving late, causing the present flight to depart late in a specific month.

- Number of delays or cancellations caused by significant meteorological conditions (actual or forecasted) that, in the judgment of the carrier, delays or prevents the operation of a flight such as tornado, blizzard or hurricane in a specific month.

- Number of delays or cancellations caused by evacuation of a terminal or concourse, re-boarding of aircraft because of security breach, inoperative screening equipment and/or long lines in excess of 29 minutes at screening areas in a specific month.

- The number of delays and cancellations attributable to the national aviation system that refer to a broad set of conditions, such as non-extreme weather conditions, airport operations, heavy traffic volume, and air traffic control in a specific month.

- The number of delays and cancellations attributable to the national aviation system that refer to a broad set of conditions, such as non-extreme weather conditions, airport operations, heavy traffic volume, and air traffic control in a specific month.

- The number of delays and cancellations due to circumstances within the airline's control (e.g. maintenance or crew problems, aircraft cleaning, baggage loading, fueling, etc.) in a specific month.

- The number of minutes delayed caused by a previous flight with the same aircraft arriving late, causing the present flight to depart late in a specific month.

- The number of minutes delayed caused by significant meteorological conditions (actual or forecasted) that, in the judgment of the carrier, delays or prevents the operation of a flight such as tornado, blizzard or hurricane in a specific month.

- The number of minutes delayed due to circumstances within the airline's control (e.g. maintenance or crew problems, aircraft cleaning, baggage loading, fueling, etc.) in a specific month.

- The number of minutes delayed caused by evacuation of a terminal or concourse, re-boarding of aircraft because of security breach, inoperative screening equipment and/or long lines in excess of 29 minutes at screening areas in a specific month.

- The number of minutes delayed attributable to the national aviation system that refer to a broad set of conditions, such as non-extreme weather conditions, airport operations, heavy traffic volume, and air traffic control in a specific month.

- The number of flights that were cancelled in a specific month.

- The number of flights that were on time in a specific month.

- The total number of flights in a specific month.

- The number of flights that were delayed in a specific month.

- *The number of flights that were diverted in a specific month.*

These information is represented in a JSON file format.

Software & Libraries

The Web framework where this Web application was based on is the Django Web Framework. The main features of the Framework that were used are as follows:

- ***Serializers***: Serializers allow complex data such as query sets and model instances to be converted to native Python datatypes that can then be easily rendered into JSON, XML or other content types. Serializers also provide deserialization, allowing parsed data to be converted back into complex types, after first validating the incoming data. They are a generic way to control the output of responses.

- ***Models***: A model is the single, definitive source of information about the data. It contains the essential fields and behaviors of the stored data. Generally, each model maps to a single database table. Each attribute of the model represents a database field. In our case, there is one big model that has fields that correspond to the information as written in the **Data** section.

- ***Views***: Views are a key component of applications built with the framework. At their simplest they are a Python function or class that takes a web request and return a web response. Views are used to do things like fetch objects from the database, modify those objects if needed, render forms, return HTML and so forth.

Because of the Django Framework, the project follows the guidelines posed by the REST Framework. More specifically, it fulfills the following:

- All resources have uniform, easily-understandable names.
- All interactions are context-free, they are independent of what has happened before.
- Components can perform only a small set of well-defined methods.
- Structure of the resources is clear (URI format).
- Each request has one representation as a response.

The whole project is based on multi-tier architecture which is a client–server architecture in which presentation, application processing, and data management functions are physically separated. This project is based on three-tier architecture.

-A front-end web server serving static content, and potentially some cached dynamic content. In web-based application, front end is the content rendered by the browser. Created with *Vue and Bootstrap.*
-A middle dynamic content processing and generation level application server.
-A back-end database or data store, comprising both data sets and the database management system software that manages and provides access to the data. Created with *ElephantSQL – PostgresSQL remote server.*

Messages & Response Codes

If the message is a GET request – meaning a request for a resource:

e.g.: GET http://127.0.0.1:8000/api/v1/airport/<code>/

The response codes would be:

- 200: OK
- 204: No Content
- 404: Not Found

If the message is a PUT or POST or DELETE request – meaning a request for alteration of a resource: :

e.g.: PUT http://127.0.0.1:8000/api/v1/airport/<code>

The response codes would be:

- 200: OK
- 201: Created
- 202: Accepted
- 204: No Content
- 400: Bad Request
- 404: Not Found