Web Engineering - Group 4

Jur Kroeze, s3254518
Derk Jan Pot, s2581566
Tom Apol, s2701650

March 2019

1 Introduction

Design document of a REST API, concerning data of US airports and their flights, between 2003/2016. The data is made available by the CORGIS dataset project.

2 Definitions

To clarify certain concepts, there is a table of explanations of concepts mentioned in the endpoints section.

- Airport code: unique string that identifies an airport (for example ATL for Atlanta, GA).
- Carrier code: unique string that identifies a carrier (for example AA for American Airlines).
- Month: a month is identified with a string of first the year, then a /, then the month. For example, 2003/6 denotes July 2003. Note that January is month 0, February is month 1, etc.
- Return: All results that will be retrieved are available in both json (return=json) and csv (return=csv) format. By default json will be returned.

3 Endpoints

1. HTTP GET `/v1/airports?return=json`
   Returns a list of all the US airports in the dataset.
   Example correct json-response: Response code: 200

```
[
    {
        "airport": {
                    "code": "PDX",
                    "name": "Portland, OR: Portland International"
                    }

    },
    {
        "airport": {
                    "code": "LGA",
                    "name": "New York, NY: LaGuardia"
                    }

    },
    ...
]
```

When the list cannot be retrieved, the response code is 503, accompanied with the message: "Error 503: List of airports cannot be found.".

2. HTTP GET `/v1/carriers?return=json`
   Returns a list of all the carriers in US airports in the dataset. Example correct json-response: response code: 200

```
[
    {
        "carrier": {
                    "code": "US",
                    "name": "US Airways Inc."
                    }

    },
    {
        "carrier": {
                    "code": "AA",
                    "name": "American Airlines Inc."
                    }

    },
    ...
]
```

When the list cannot be retrieved, we the response code is 503, accompanied with the message: "Error 503: List of carriers cannot be found.".

3. HTTP GET `/v1/carriers?airport-code=&return=json`
   response code: 200
   Returns all carriers operating at a given airport, given by an airport code.
   Example correct response: response code: 200

```
[
    {
        "carrier": {
                    "code": "US",
                    "name": "US Airways Inc."
                    }

    },
    {
        "carrier": {
                    "code": "AA",
                    "name": "American Airlines Inc."
                    }

    },
    ...
]
```

   When given an invalid carrier-code (i.e. one that is not in the database) we return status code 400.

4. (a) HTTP GET `/v1/carriers/statistics/{carrier-code}/flights?airport-code=&year=&month=&return=json`
   Returns all statistics about flights of a carrier identified by a carrier code, on a given airport identified by an airport code, on a given month of a given year. If no month and year is given, it will return the data for all months. Example correct response: response code: 200

```
    [
        {
            "flights": {
                        "cancelled": 5,
                        "on time": 561,
                        "total": 752,
                        "delayed": 186,
                        "diverted": 0
                        },
            "airport": {
                        "code": "ATL",
```

```
                    "name": "Atlanta, GA: Hartsfield-Jackson Atlanta
                    },
            "carrier": {
                    "code": "AA",
                    "name": "American Airlines Inc."
                    },
            "time": {
                    "2003/06",
                    "year": 2003,
                    "month": 6
                    }

        }
    ]
```


When given an invalid carrier-code (i.e. one that is not in the database) we return status code 400.

(b) HTTP PUT `/v1/carriers/statistics/{carrier-code}/flights?airport-code=&month=` Modifies one or more statistics about flights of a carrier identified by a carrier code, on a given airport identified by an airport code, on a given month. Example correct response:
response code: 204
When given an invalid carrier-code (i.e. one that is not in the database) we return status code 400.

(c) HTTP POST `/v1/carriers/statistics/{carrier-code}/flights?airport-code=&month=`
Creates one or more statistics about flights of a carrier identified by a carrier code, on a given airport identified by an airport code, on a given month. Example correct response code:
response code: 201
When given an invalid carrier-code (i.e. using too few or too many characters, or a carrier-code that is already in the database) we return status code 400.

(d) HTTP DELETE `/v1/carriers/statistics/{carrier-code}/flights?airport-code=&month=`
response code: 204
Requests the deletion of one or more statistics about flights of a carrier identified by a carrier code, on a given airport id, for a given month. If no month was specified, the specified statistics (given the carrier code and airport code) for all months are requested for deletion.
When given an invalid carrier-code (i.e. one that is not in the database) we return status code 400.

5. HTTP GET `/v1/carriers/statistics/{carrier-code}/flights/delays?airport-code=&month=&year=&return=json`

response code: 200
Returns the number of on-time, delayed and cancelled flights of a given carrier identified by a carrier code from/to a given airport identified with an airport code. Optionally, a month and year can be specified. Example response: response code 200


```
[
    {
        "delays": {
                    "late aircraft": 18,
                    "weather": 28,
                    "security": 2,
                    "national aviation system": 105,
                    "carrier": 34
                    }

    },
    ...
]
```


When given an invalid carrier-code (i.e. one that is not in the database) we return status code 400.

6. HTTP GET `/v1/carriers/statistics/{carrier-code}/delay-minutes?reason=&airport-code=&month=&year=&return=json`
response code: 200
Returns the number of minutes of delay per carrier, identified by a carrier code. If the reason specified is "carrier-specific", it will only return the amount of minutes that can be attributed to the carrier. Similarly for the reasons "weather", "security", "late aircraft", and "national aviation system". If no reason is specified, it will return the total amount of minutes of delay of the given carrier plus the individual delays. Optionally, a month and year can be specified to get only the delays for one specific month. Example response for with reason "carrier: response code: 200


```
[
    "[1367, 4201, 1058, 968, 2048, 2098, 31999, 1029, 2092, 1765, 997, 8
]
```


Example response with no reason specified:


```
[
    {
        "time":    {
                    "year": 2003,
                    "month": 4,
```

```
            }
            "delays": {
                        "late aircraft": 1269,
                        "weather": 1722,
                        "security": 139,
                        "national aviation system": 3817,
                        "carrier": 1367,
                        "total": 8314
                        }
        },
        ...
]
```

When given an invalid carrier-code (i.e. one that is not in the database) we return status code 400.

7. HTTP GET `/v1/airports/statistics/descriptive?airport-a=&airport-b=&carrier-code=&return=json`
   response code: 200
   Return the mean, median and standard deviation for carrier-specific delays for a route, for a specific carrier. The route is denoted by the two respective airports, between airport A and airport B. (Note that it is unknown which airport is the starting point and which is the destination.) These airports are both denoted by their airport codes.

   Example correct json-response: response code: 200

```
[
    {
        "carrier-code": "AA",
        "airport-a": "PDX",
        "airport-b": "BOS",
        "stats": {
            "mean": 7,
            "median": 8,
            "standard-deviation": 10
            }
    },
    ...
]
```

   If given an airport-airline combination that does not occur in the database will result in response code 404, accompanied with the message: "Error 404: the given airline does not fly on one or both of the given airports".

4 Back-end

For the back-end we used the Django framework, because it is really easy to use and quick to set up. Our data will be stored in a sqlite database, since it's the default type of database used by Django. Since there are two main endpoints of our API (airport and carrier data), we divided our program as such. We define urlpatterns for the different functionalities, which are handled by simple python functions. These retrieve the data from the database by looping through the database for specific airport/carrier codes and possibly other arguments specified by the function-caller.

5 Front-end

If you would like to get a more visual representation of what this API can do, you can also go to `/` (so just the main domain, no subdomains) and from there navigate a front end site that can interact with this API. You will be able to test all the query's and see what they return. There is a page for each of the query's listed in the document. You can find them in the navigation bar at the top of the screen. The front-end was implemented using Django in combination with css and html.