# Web Engineering Project Documentation

Dimitris Aktsoglou (s2979616), Emanuel Nae (s2931931), Daan Groot (s2958287)

March 21, 2019

## 1 Introduction

Our aim is to create an API that is able to collect and distribute data on demand for all USA airports and carriers. The data includes names and id of both airport and carriers as well as well as statistics that have been recorder over a period of time for these two parameters. These statistics should be able to represent an overview over a carrier or an airport. Statistics include,but not limited to, number of delays for a carrier for a number of reasons , delay times (in minutes) for all reason as well as the total amount, number of delays for an airport e.t.c.

The decisions we make in the design of our API are based on RESTful Design principles. REST design is in general based upon the following principles:

1. Resources, which are any kind of object, data, or service that can be accessed by the client.
2. Resource has an identifier, which is a URI that uniquely identifies that resource.
3. Client interacts with a service by exchanging representation of resources.
4. Use a uniform interface, which helps to decouple the client and service implementations.

Furthermore there are four level of "Maturity" that have been defined in such a design. Our goal is to design an API as close as possible to Maturity level 3.

## 2 Milestone 1: API Design

### 2.1 Resources

Resources refers to the information that can be returned by an API. In our case, the various resources that can be returned are the following:

1. **Airports object**: represents all the airports that are available in the USA, which has parameters such as `code` and `name`.
2. **Statistics object**: returns statistics about flights. It has the following parameters: `flights`, `number of delays` and `minutes delayed`.
   - 2.1. **Flights object**: returns information about the number of `cancelled`, `on time`, `delayed`, `diverted` flights and their `total`.
   - 2.2. **Number of delays object**: returns the number of delays categorized by reasons such as: `late aircraft`, `weather`, `security`, `national aviation system` and `carrier`.

2.3. **Minutes delayed object**: returns the number of minutes delayed per reason of delay: `late aircraft, weather, security, national aviation system` and `carrier`.

3. **Time object**: returns information about time based on the following parameters: `year, month` and `label`

4. **Carriers object**: returns the carriers that have flights in the USA, which have a `code` and a `name` parameter.

## 2.2 JSON Hyper-Schema

JSON Hyper-Schema is a JSON Schema vocabulary that allows the user to annotate JSON documents with hyperlinks and instructions for processing and manipulating remote JSON resources through hypermedia environments. In our case, since we are using REST API, the hypermedia environment that is being used is HTTP.

1. **Airports object**

```
{
    "type": "object",
    "properties": {
        "code": {
            "type": "string",
            "readOnly": true
        }
        "name": {
            "type": "string",
            "readOnly": true
        }
    },
    "links": [
        {
            "rel": "self",
            "href": "airports/{airport_code}"
        }
    ],
    "required": ["code"]
}
```

2. **Statistics object**

```
{
    "type": "object",
    "properties": {
        "flights": {
            "type": "string",
            "readOnly": true
        }
        "# of delays": {
            "type": "string",
            "readOnly": true
```

```
        }
        "minutes delayed": {
            "type": "string",
            "readOnly": true
        }
    },
    "links": [
        {
            "rel": "airports", "carriers",
            "href": "airports/carriers/stats/flights"
        }
    ]
}
```

2.1. **Flights object**:

```
    {
        "type": "object",
        "properties": {
            "cancelled": {
                "type": "number",
            }
            "on time": {
                "type": "number",
            }
            "converted": {
                "type": "number",
            }
            "delayed": {
                "type": "number",
            }
            "total": {
                "type": "number",
            }
        },
        "links": [
            {
                "rel": "stats",
                "href": "airports/carriers/stats/flights"
            }
        ]
    }
```

2.2. **Number of delays object**

```
    {
        "type": "object",
        "reasons": {
            "late aircraft": {
                "type": "number",
            }
```

```
            "weather": {
                "type": "number",
            }
            "security": {
                "type": "number",
            }
            "national aviation system": {
                "type": "number",
            }
            "carrier": {
                "type": "number",
            }
        },
        "links": [
            {
                "rel": "stats",
                "href": "airports/carriers/stats/#ofdelays"
            }
        ]
    }
```

2.3. **Minutes delayed object**:

```
    {
        "type": "object",
        "reasons": {
            "late aircraft": {
                "type": "number",
            }
            "weather": {
                "type": "number",
            }
            "security": {
                "type": "number",
            }
            "national aviation system": {
                "type": "number",
            }
            "carrier": {
                "type": "number",
            }
            "total": {
                "type": "number",
            }
        },
        "links": [
            {
                "rel": "stats",
                "href": "airports/carriers/stats/minutesdelayed"
            }
        ]
```

```
    }
```

3. **Time object**:

```
{
    "type": "object",
    "properties": {
        "label": {
            "type": "string",
            "readOnly": true
        }
        "year": {
            "type": "number",
            "readOnly": true
        }
        "month": {
            "type": "number",
            "readOnly": true
        }
    },
    "links": [
        {
            "rel": "airports", "carriers", "stats",
            "href": "airports/carriers/stats/time?year={year}&month={month}"
        }
    ],
    "required": ["year","month"]
}
```

4. **Carriers object**:

```
{
    "type": "object",
    "properties": {
        "code": {
            "type": "string",
            "readOnly": true
        }
        "name": {
            "type": "string",
            "readOnly": true
        }
    },
    "links": [
        {
            "rel": "self",
            "href": "carriers\{carrier_code}"
        }
    ],
    "required": ["code"]
}
```

## 2.3 Endpoints

Endpoints represent the entries that give access to the resource. Currently, the endpoints presented below refer strictly to the ones given in the Project Description document, but they may be modified in the future.

1. **GET /airports**
   Returns all airports available in the US.

   Request headers:

   - `Accept: application/json`
   - `Accept: text/csv`

   Request body:

   `empty`

   Response headers:

   - `Content-Type: application/json`
   - `Content-Type: text/csv`

   Responses with content:

   - 200 OK Successful Operation
     `: array[string]  The 3-letter codes of the airports.`
   - 5xx Internal Server Error
     `empty`

2. **GET /carriers**
   Returns all carriers operating in US airports.

   Request headers:

   - `Accept: application/json`
   - `Accept: text/csv`

   Request body:

   `empty`

   Response headers:

   - `Content-Type: application/json`
   - `Content-Type: text/csv`

   Responses with content:

   - 200 OK Successful Operation

: array[string]  The 3-letter codes of the carriers.

- 5xx Internal Server Error
  empty

3. **GET /airports/{airport_code}/carriers**
   Returns all carriers operating at a specific US airport.

   Endpoint path parameters:

   - airport_code  The 3-letter code of this airport.

   Request headers:

   - Accept: application/json
   - Accept: text/csv

   Request body:

   empty

   Response headers:

   - Content-Type: application/json
   - Content-Type: text/csv

   Responses with content:

   - 200 OK Successful operation
     : array[string]  The 3-letter codes of the carriers.
   - 400 Bad Request
     empty
   - 401 Unauthorized
     empty
   - 404 Not Found
     empty
   - 5xx Internal Server Error
     empty

4. **GET /airports/{airport_code}/carriers/{carrier_code}/stats?year={year}&month={month}**
   Returns statistics about flights of a carrier from/to an US airport for a given year and month.

   **GET /airports/{airport_code}/carriers/{carrier_code}/stats**
   Returns statistics about flights of a carrier from/to an US airport for all months available.

   Endpoint path parameters:

   - airport_code  The 3-letter code of the airport.

- `carrier_code` The 3-letter code of the carrier.
- `year` The year for which data should be returned.
- `month` The month for which data should be returned.

Request headers:

- `Accept: application/json`
- `Accept: text/csv`

Request body:

`empty`

Response headers:

- `Content-Type: application/json`
- `Content-Type: text/csv`

Responses with content:

- 200 OK Successful operation
  /airports/{airport_code}/carriers/{carrier_code}/stats?year={year}&month={month}

```
flights: object
    cancelled: int
    on-time: int
    total: int
    delayed: int
    diverted: int
delays-counts: object
    late-aircraft: int
    weather: int
    security: int
    national-aviation-system: int
    carrier: int
minutes-delayed: object
    late-aircraft: int
    weather: int
    security: int
    national-aviation-system: int
    carrier: int
    total: int
```

/airports/{airport_code}/carriers/{carrier_code}/stats

```
: array[object]
    flights: object
        cancelled: int
        on-time: int
        total: int
        delayed: int
```

```
        diverted: int
    delays-counts: object
        late-aircraft: int
        weather: int
        security: int
        national-aviation-system: int
        carrier: int
    minutes-delayed: object
        late-aircraft: int
        weather: int
        security: int
        national-aviation-system: int
        carrier: int
        total: int
    time: object
        year: int
        month: int
```

- 400 Bad Request

  `empty`
- 401 Unauthorized

  `empty`
- 404 Not Found

  `empty`
- 5xx Internal Server Error

  `empty`

**POST /airports/carriers/stats**
Add statistics about flights of carriers from/to US airports.

Request headers:

- `Content-Type: application/json`
- `Content-Type: text/csv`

Request body:

```
: array[object]
    airport: object
        code: string
        name: string
    carrier: object
        code: string
        name: string
    yearMonth: object
        year: int
        month: int
```

```
statisticData: object
    cancelledFlightCount: int
    cancelledFlightCount: int
    delayedFlightCount: int
    divertedFlightCount: int
    totalFlightCount: int
    lateAircraftDelayCount: int
    carrierDelayCount: int
    weatherDelayCount: int
    securityDelayCount: int
    nationalAviationSystemDelayCount: int
    lateAircraftDelayTime: int
    carrierDelayTime: int
    weatherDelayTime: int
    securityDelayTime: int
    nationalAviationSystemDelayTime: int
    totalDelayTime: int
```

Response headers:

```
empty
```

Responses with content:

- 200 OK Successful operation
  `empty`
- 400 Bad Request
  `empty`
- 401 Unauthorized
  `empty`
- 404 Not Found
  `empty`
- 5xx Internal Server Error
  `empty`

**PUT /airports/{airport_code}/carriers/{carrier_code}/stats?year={year}&month={month}**
Update statistics about flights of a carrier from/to an US airport for a given year and month.

**PUT /airports/{airport_code}/carriers/{carrier_code}/stats**
Update statistics about flights of a carrier from/to an US airport for all months available.

Endpoint path parameters:

- `airport_code` The 3-letter code of the airport.
- `carrier_code` The 3-letter code of the carrier.
- `year` The year for which data should be returned.
- `month` The month for which data should be returned.

Request headers:

- `Content-Type: application/json`
- `Content-Type: text/csv`

Request body:

/airports/{airport_code}/carriers/{carrier_code}/stats?year={year}&month={month}

```
flights: object
    cancelled: int
    on-time: int
    total: int
    delayed: int
    diverted: int
delays-counts: object
    late-aircraft: int
    weather: int
    security: int
    national-aviation-system: int
    carrier: int
minutes-delayed: object
    late-aircraft: int
    weather: int
    security: int
    national-aviation-system: int
    carrier: int
    total: int
```

/airports/{airport_code}/carriers/{carrier_code}/stats

```
: array[object]
    flights: object
        cancelled: int
        on-time: int
        total: int
        delayed: int
        diverted: int
    delays-counts: object
        late-aircraft: int
        weather: int
        security: int
        national-aviation-system: int
        carrier: int
    minutes-delayed: object
        late-aircraft: int
        weather: int
        security: int
        national-aviation-system: int
        carrier: int
        total: int
    time: object
```

```
            year: int
            month: int
```

Response headers:

```
empty
```

Responses with content:

- 200 OK Successful operation
  ```
  empty
  ```
- 400 Bad Request
  ```
  empty
  ```
- 401 Unauthorized
  ```
  empty
  ```
- 404 Not Found
  ```
  empty
  ```
- 5xx Internal Server Error
  ```
  empty
  ```

**DELETE /airports/{airport_code}/carriers/{carrier_code}/stats?year={year}&month={month}**
Delete statistics about flights of a carrier from/to an US airport for a given year and month.

**DELETE /airports/{airport_code}/carriers/{carrier_code}/stats**
Delete statistics about flights of a carrier from/to an US airport for all months available.

Endpoint path parameters:

- `airport_code` The 3-letter code of the airport.
- `carrier_code` The 3-letter code of the carrier.
- `year` The year for which data should be returned.
- `month` The month for which data should be returned.

Request headers:

- `Content-Type: application/json`
- `Content-Type: text/csv`

Request body:
/airports/{airport_code}/carriers/{carrier_code}/stats?year={year}&month={month}

```
empty
```

/airports/{airport_code}/carriers/{carrier_code}/stats

```
: array[object]
    time: object
        year: int
        month: int
```

Response headers:

`empty`

Responses with content:

- 200 OK Successful operation
  `empty`
- 400 Bad Request
  `empty`
- 401 Unauthorized
  `empty`
- 404 Not Found
  `empty`
- 5xx Internal Server Error
  `empty`

5. **GET /airports/{airport_code}/carriers/{carrier_code}/stats/flights?year={year}&month={month}**
   Returns the number of on-time, delayed, and cancelled flights of a carrier from/to an US airport for a given year and month.

   **GET /airports/{airport_code}/carriers/{carrier_code}/stats/flights**
   Returns the number of on-time, delayed, and cancelled flights of a carrier from/to an US airport for all months available.

   Endpoint path parameters:

   - `airport_code` The 3-letter code of the airport.
   - `carrier_code` The 3-letter code of the carrier.
   - `year` The year for which data should be returned.
   - `month` The month for which data should be returned.

   Request headers:

   - `Accept: application/json`
   - `Accept: text/csv`

   Request body:

   `empty`

   Response headers:

   - `Content-Type: application/json`
   - `Content-Type: text/csv`

   Responses with content:

- 200 OK Successful operation
  /airports/{airport_code}/carriers/{carrier_code}/stats/flights?year={year}&month={month}
  ```
  on time: int
  delayed: int
  cancelled: int
  ```
  /airports/{airport_code}/carriers/{carrier_code}/stats/flights
  ```
  : array[object]
      on-time: int
      delayed: int
      cancelled: int
      time: object
          year: int
          month: int
  ```
- 400 Bad Request
  ```
  empty
  ```
- 401 Unauthorized
  ```
  empty
  ```
- 404 Not Found
  ```
  empty
  ```
- 5xx Internal Server Error
  ```
  empty
  ```

6. **GET /airports/{airport_code}/carriers/stats/delay-times?reason=carrier,late-aircraft &year={year}&month={month}**
   Returns the number of minutes of delay per carrier attributed to carrier-specific reasons, for a given year and month and for a specific airport.

   **GET /airports/{airport_code}/carriers/stats/delay-times?year={year}&month={month}**
   Returns the number of minutes of delay per carrier attributed to all reasons, for a given year and month and for a specific airport.

   **GET /airports/carriers/stats/delay-times?reason=carrier,late-aircraft&year={year} &month={month}**
   Returns the number of minutes of delay per carrier attributed to carrier-specific reasons, for a given year and month and for all US airports.

   **GET /airports/carriers/stats/delay-times?year={year}&month={month}**
   Returns the number of minutes of delay per carrier attributed to all reasons, for a given year and month and for all US airports.

   **GET /airports/{airport_code}/carriers/stats/delay-times?reason=carrier,late-aircraft**
   Returns the number of minutes of delay per carrier attributed to carrier-specific reasons, for all months available and for a specific airport.

   **GET /airports/{airport_code}/carriers/stats/delay-times**
   Returns the number of minutes of delay per carrier attributed to all reasons, for all months

available and for a specific airport.

**GET /airports/carriers/stats/delay-times?reason=carrier,late-aircraft**
Returns the number of minutes of delay per carrier attributed to carrier-specific reasons, for all months available and for all US airports.

**GET /airports/carriers/stats/delay-times**
Returns the number of minutes of delay per carrier attributed to all reasons, for all months available and for all US airports.

Endpoint path parameters:

- `airport_code` The 3-letter code of the airport.
- `carrier_code` The 3-letter code of the carrier.
- `year` The year for which data should be returned.
- `month` The month for which data should be returned.

Request headers:

- `Accept: application/json`
- `Accept: text/csv`

Request body:

`empty`

Response headers:

- `Content-Type: application/json`
- `Content-Type: text/csv`

Responses with content:

- 200 OK Successful operation
  /airports/{airport_code}/carriers/stats/delay-times?reason=carrier,late-aircraft&year={year}&month={month}

```
: array[object]
    minutes-delayed: object
        carrier: int
        late-aircraft: int
    carrier: object
        code: string
```

  /airports/{airport_code}/carriers/stats/delay-times?year={year}&month={month}

```
: array[object]
    minutes-delayed: object
        late-aircraft: int
        weather: int
        carrier: int
        security: int
        national-aviation-system: int
    carrier: object
        code: string
```

/airports/carriers/stats/delay-times?reason=carrier,late-aircraft&year={year}&month={month}

```
: array[object]
    carrier-delays: array[object]
        minutes-delayed: object
            carrier: int
            late-aircraft: int
        carrier: object
            code: string
    airport: object
        code: string
```

/airports/carriers/stats/delay-times?&year={year}&month={month}

```
: array[object]
    carrier-delays: array[object]
        minutes-delayed: object
            late-aircraft: int
            weather: int
            carrier: int
            security: int
            national-aviation-system: int
        carrier: object
            code: string
    airport: object
        code: string
```

/airports/{airport_code}/carriers/stats/delay-times?reason=carrier,late-aircraft

```
: array[object]
    : array[object]
        minutes-delayed: object
            carrier: int
            late-aircraft: int
        carrier: object
            code: string
    time: object
        year: int
        month: int
```

/airports/{airport_code}/carriers/stats/delay-times

```
: array[object]
    : array[object]
```

```
        minutes-delayed: object
            late-aircraft: int
            weather: int
            carrier: int
            security: int
            national-aviation-system: int
        carrier: object
            code: string
    time: object
        year: int
        month: int
```

/airports/carriers/stats/delay-times?reason=carrier,late-aircraft

```
: array[object]
    : array[object]
        carrier-delays: array[object]
            minutes-delayed: object
                carrier: int
                late-aircraft: int
            carrier: object
                code: string
        airport: object
            code: string
    time: object
        year: int
        month: int
```

/airports/carriers/stats/delay-times

```
: array[object]
    : array[object]
        carrier-delays: array[object]
            minutes-delayed: object
                late-aircraft: int
                weather: int
                carrier: int
                security: int
                national-aviation-system: int
            carrier: object
                code: string
        airport: object
            code: string
    time: object
        year: int
        month: int
```

- 400 Bad Request

  ```
  empty
  ```

- 401 Unauthorized

  ```
  empty
  ```

- 404 Not Found
  `empty`
- 5xx Internal Server Error
  `empty`

7. **GET /airports/{airport_code_1}/{airport_code_2}/carriers/{carrier_code}/extra-stats/delay-times**
   Returns the descriptive statistics, mean, median, and standard deviation, for carrier-specific delays for a flight between any two airports in the USA for a specific carrier serving this route.

   **GET /airports/{airport_code_1}/{airport_code_2}/carriers/extra-stats/delay-times**
   Returns the descriptive statistics, mean, median, and standard deviation, for carrier-specific delays for a flight between any two airports in the USA for all carriers serving this route.

   Endpoint path parameters:

   - `airport_code_1` `The 3-letter code of the first airport.`
   - `airport_code_2` `The 3-letter code of the second airport.`
   - `carrier_code` `The 3-letter code of the carrier.`

   Request headers:

   - `Accept: application/json`
   - `Accept: text/csv`

   Request body:

   `empty`

   Response headers:

   - `Content-Type: application/json`
   - `Content-Type: text/csv`

   Responses with content:

   - 200 OK Successful operation
     /airports/{airport_code_1}/{airport_code_2}/carriers/{carrier_code}/extra-stats/delay-times

     ```
     carrier: object
         mean: int
         median: int
         standard-deviation: int
     late-aircraft: object
         mean: int
         median: int
         standard-deviation: int
     ```

     /airports/{airport_code_1}/{airport_code_2}/carriers/extra-stats/delay-times

```
: array[object]
    carrier-delay: object
        mean: int
        median: int
        standard-deviation: int
    late-aircraft: object
        mean: int
        median: int
        standard-deviation: int
    carrier: object
        code: string
```

- 400 Bad Request
  `empty`
- 401 Unauthorized
  `empty`
- 404 Not Found
  `empty`
- 5xx Internal Server Error
  `empty`

# 3 Server Responses Data

## 3.1 Data

The requested data will be served in JSON format. Next follow some examples of data in JSON returned by the server for a **GET** request to each endpoint:

1. **GET** request to:
   `/airports`
   JSON response example:
   `[ATL, CLT, SAN]`
   CSV response example:

| ATL |
|---|
| CLT |
| SAN |

2. **GET** request to:
   `/carriers`
   JSON response example:
   `[CO, VX, SAN]`
   CSV response example:

| CO |
|---|
| VX |
| SAN |

3. **GET** request to:

   `/airports/{airport_code}/carriers`

   JSON response example:

   `[CO, VX, SAN]`

   CSV response example:

| CO |
|---|
| VX |
| SAN |

4. **GET** request to:

   `/airports/{airport_code}/carriers/{carrier_code}/stats?year={year}&month={month}`

   JSON response example:

```
{
    "flights": {
        "cancelled": 5,
        "on time": 561,
        "total": 752,
        "delayed": 186,
        "diverted": 0
    },
    "# of delays": {
        "late aircraft": 18,
        "weather": 28,
        "security": 2,
        "national aviation system": 105,
        "carrier": 34
    },
    "minutes delayed": {
        "late aircraft": 1269,
        "weather": 1722,
        "carrier": 1367,
        "security": 139,
        "total": 8314,
        "national aviation system": 3817
    }
}
```

   CSV response example:

| Cancelled | OnTIme | Total | Delayed | Diverted |
|---|---|---|---|---|
| 5 | 561 | 752 | 186 | 0 |

| # of Delay(late aircraft) | # of Delay(weather) | # of Delay(security) | # of Delay(N.A.S) | # of Delay(carrier) |
|---|---|---|---|---|
| 18 | 28 | 2 | 105 | 34 |

| Min-Delayed(late aircraft) | Min-Delayed(weather) | Min-Delayed(carrier) | Min-Delayed(security) | Min-Delayed(total) |
|---|---|---|---|---|
| 1269 | 1722 | 1367 | 139 | 8314 |

**GET** request to:

`/airports/{airport_code}/carriers/{carrier_code}/stats`

JSON response example:


```
[
    {
        "flights": {
            "cancelled": 5,
            "on time": 561,
            "total": 752,
            "delayed": 186,
            "diverted": 0
        },
        "# of delays": {
            "late aircraft": 18,
            "weather": 28,
            "security": 2,
            "national aviation system": 105,
            "carrier": 34
        },
        "minutes delayed": {
            "late aircraft": 1269,
            "weather": 1722,
            "carrier": 1367,
            "security": 139,
            "total": 8314,
            "national aviation system": 3817
        },
        "time": {
            "label": "2003/6",
            "year": 2003,
            "month": 6
        }
    },
    {
        "flights": {
            "cancelled": 7,
            "on time": 1034,
            "total": 1266,
            "delayed": 225,
            "diverted": 0
        },
        "# of delays": {
            "late aircraft": 46,
```

```
            "weather": 24,
            "security": 2,
            "national aviation system": 84,
            "carrier": 69
        },
        "minutes delayed": {
            "late aircraft": 3043,
            "weather": 1783,
            "carrier": 4201,
            "security": 45,
            "total": 12139,
            "national aviation system": 3067
        },
        "time": {
              "label": "2003/10",
              "year": 2003,
              "month": 10
        }
    }
]
```


5. **GET** request to:

   `/airports/{airport_code}/carriers/{carrier_code}/stats/flights?year={year}&month={month}`

   JSON response example:


```
{
    "on time": 561,
    "delayed": 186,
    "cancelled": 5
}
```


   **GET** request to:

   `/airports/{airport_code}/carriers/{carrier_code}/stats/flights`

   JSON response example:


```
[
    {
        "on time": 561,
        "delayed": 186,
        "cancelled": 5,
        "time": {
            "label": "2003/6",
            "year": 2003,
            "month": 6
        }
    },
    {
        "on time": 1034,
        "delayed": 225,
        "cancelled": 7,
```

```
        "time": {
              "label": "2003/10",
              "year": 2003,
              "month": 10
        }
    }
]
```

6. **GET** request to:

   `/airports/{airport_code}/carriers/stats/delay-times?reason=carrier,late-aircraft&year={year}&month={month}`

   JSON response example:

```
[
    {
        "minutes delayed": {
            "carrier": 1367,
            "late aircraft": 1269
        },
        "carrier": {
            "code": "AA",
            "name": "American Airlines Inc."
        }
    },
    {
        "minutes delayed": {
            "carrier": 69,
            "late aircraft": 277
        },
        "carrier": {
            "code": "XE",
            "name": "ExpressJet Airlines Inc."
        }
    }
]
```

   **GET** request to:

   `/airports/carriers/stats/delay-times?reason=carrier,late-aircraft&year={year}&month={month}`

   JSON response example:

```
[
    {
        "carrier-delays": [
            {
                "minutes delayed": {
                    "carrier": 1367,
                    "late aircraft": 1269
                },
                "carrier": {
                    "code": "AA",
```

```
                    "name": "American Airlines Inc."
                }
            },
            {
                "minutes delayed": {
                    "carrier": 69,
                    "late aircraft": 277
                },
                "carrier": {
                    "code": "XE",
                    "name": "ExpressJet Airlines Inc."
                }
            }
        ],
        "airport": {
          "code": "ATL",
          "name": "Atlanta, GA: Hartsfield-Jackson Atlanta International"
        }
    },
    {
        "carrier-delays": [
            {
                "minutes delayed": {
                    "carrier": 5,
                    "late aircraft": 0
                },
                "carrier": {
                    "code": "AA",
                    "name": "American Airlines Inc."
                }
            },
            {
                "minutes delayed": {
                    "carrier": 690054684655,
                    "late aircraft": 15699900
                },
                "carrier": {
                    "code": "AL",
                    "name": "AlwaysLate Airlines Inc."
                }
            }
        ],
        "airport": {
          "code": "SEA",
          "name": "Seattle, WA: Seattle/Tacoma International"
        }
    }
]
```


**GET** request to:

`/airports/{airport_code}/carriers/stats/delay-times?year={year}&month={month}`

JSON response example:


```
[
    {
        "minutes delayed": {
            "late aircraft": 775,
            "weather": 155,
            "carrier": 1478,
            "security": 0,
            "national aviation system": 3343
        },
        "carrier": {
            "code": "AA",
            "name": "American Airlines Inc."
        }
    },
    {
        "minutes delayed": {
            "late aircraft": 456,
            "weather": 658,
            "carrier": 1414,
            "security": 2,
            "national aviation system": 3543
        },
        "carrier": {
            "code": "XE",
            "name": "ExpressJet Airlines Inc."
        }
    }
]
```


**GET** request to:

`/airports/carriers/stats/delay-times?year={year}&month={month}`

JSON response example:


```
[
    {
        "carrier-delays": [
            {
                "minutes delayed": {
                    "late aircraft": 775,
                    "weather": 155,
                    "carrier": 1478,
                    "security": 0,
                    "national aviation system": 3343
                },
                "carrier": {
                    "code": "AA",
                    "name": "American Airlines Inc."
```

```
            }
        },
        {
            "minutes delayed": {
                "late aircraft": 123,
                "weather": 321,
                "carrier": 213,
                "security": 312,
                "national aviation system": 0
            },
            "carrier": {
                "code": "XE",
                "name": "ExpressJet Airlines Inc."
            }
        }
    ],
    "airport": {
      "code": "ATL",
      "name": "Atlanta, GA: Hartsfield-Jackson Atlanta International"
    }
},
{
    "carrier-delays": [
        {
            "minutes delayed": {
                "late aircraft": 721,
                "weather": 462,
                "carrier": 5555,
                "security": 10,
                "national aviation system": 3343
            },
            "carrier": {
                "code": "AA",
                "name": "American Airlines Inc."
            }
        },
        {
            "minutes delayed": {
                "late aircraft": 0,
                "weather": 0,
                "carrier": 0,
                "security": 1000000000,
                "national aviation system": 0
            },
            "carrier": {
                "code": "AL",
                "name": "AlwaysLate Airlines Inc."
            }
        }
```

```
        ],
        "airport": {
          "code": "SEA",
          "name": "Seattle, WA: Seattle/Tacoma International"
        }
    }
]
```


7. **GET** request to:

   `/airports/{airport_code_1}/{airport_code_2}/carriers/{carrier_code}/extra-stats/delay-times`

   JSON response example:


```
{
    "carrier": {
        "mean": 775,
        "median": 155,
        "standard deviation": 1478
    },
    "late aircraft": {
        "mean": 775,
        "median": 1554,
        "standard deviation": 1235
    }
}
```


   **GET** request to:

   `/airports/{airport_code_1}/{airport_code_2}/carriers/extra-stats/delay-times`

   JSON response example:

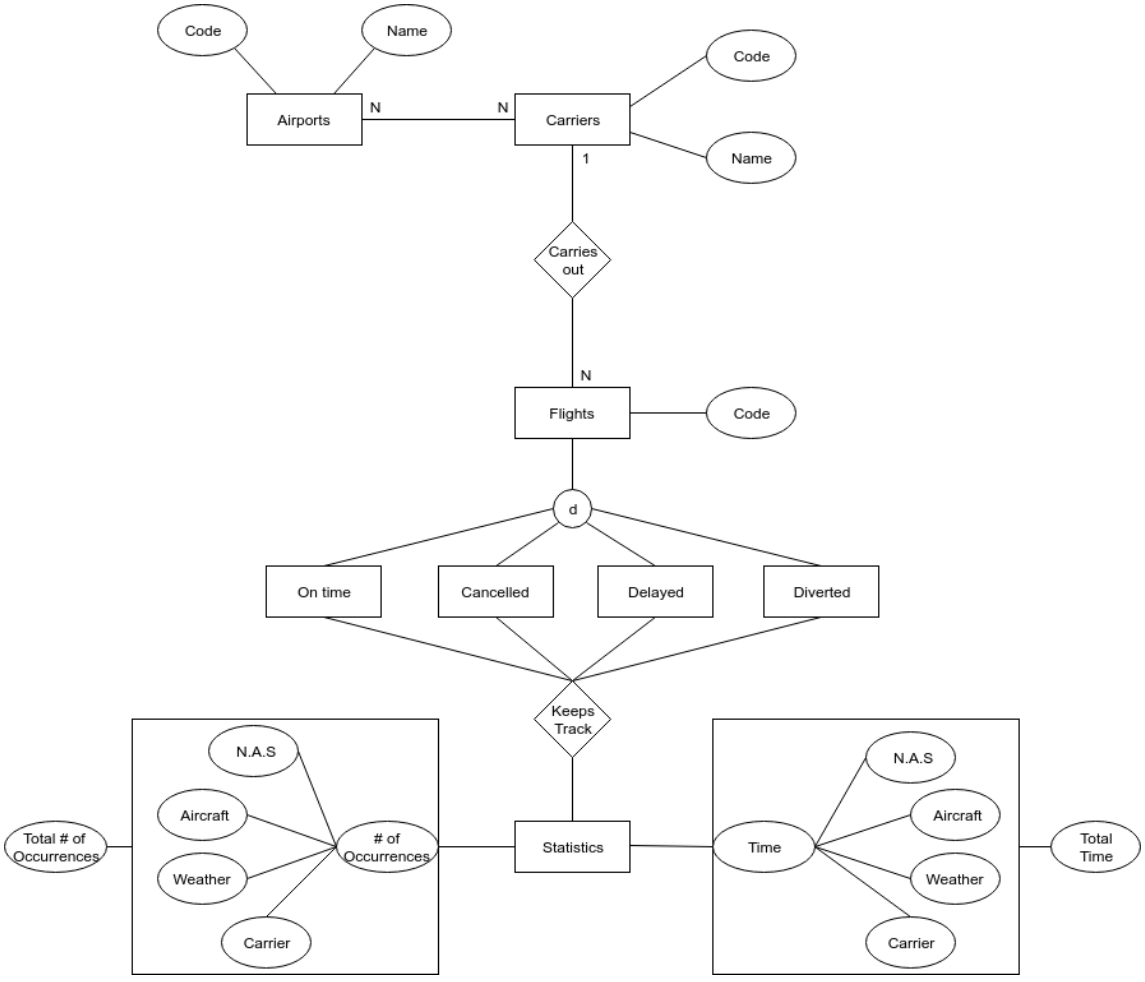

```
[
    {
        "carrier delay": {
            "mean": 775,
            "median": 155,
            "standard deviation": 1478
        },
        "late aircraft": {
            "mean": 775,
            "median": 1554,
            "standard deviation": 1235
        },
        "carrier": {
            "code": "DL",
            "name": "Delta Air Lines Inc."
        }
    },
    {
        "carrier delay": {
            "mean": 775,
            "median": 155,
```

```
            "standard deviation": 1478
        },
        "late aircraft": {
            "mean": 775000,
            "median": 1589,
            "standard deviation": 1458
        },
        "carrier": {
            "code": "AL",
            "name": "AlwaysLate Airlines Inc."
        }
    }
]
```


# 4 ER Model

This is an alpha version of our ER model. We identify three main entities in our model: **Airports, Carries and Flights**. The first two entities are pretty straight forward. Each of contains a name and a code as their identifiers. For the third entity things are a bit more complicated. Flights are mainly identified by the flight code. However we also need to keep track of the statistics of all flights. To do that we decided to split the Flight entity into 4 sub-entities. **On-time, Cancelled Delayed and Diverted.** For each of those sub categories we keep track of the statistics. We have a large number of statistics. First of all we need to keep track the reason a flight has been delayed due to different reasons( N.A.S,aircraft,weather,carrier),which are provided to us in both json and csv files. Secondly we also need to keep track of the total time(in minutes) of each of those reason separately and the combined time of all those reasons.

Since this is an Alpha version of our ER model, some of the entities and identifiers might change in the future. However we feel its important to provide a starting point for a visual representation of our thought process.




# 5 Tecnhology Stack

1. **RESTful API**. Services provide interoperability between computer systems on the Internet. REST-compliant web services allow the requesting systems to access and manipulate textual representations of web resources by using a uniform and predefined set of stateless operations. The framework we used is called **Spring**.

2. **Json** (JavaScript Object Notation). JSON is a syntax for storing and exchanging data. JSON is text, written with JavaScript object notation.

3. **Maven**. Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.

4. MySQL is an open source relational database management system. It was an obvious choice for us as it is easy to use and can manage large quantities of data.

# 6 Project Structure

THe structure of our code is devided into four parts: The converters , the models, the database and the rest application (main).

1. **Converters**.

   Since we need the ability to create and exhange data in both json and csv formats we created two different java files, JsonConverter and CSVconverter. Each of those is able to handle serialization and deserialization of their corresponding formats. In the case of Json we decided to use **Gson** library as it is reliable and easy to use, since it uses reflection so it does not require additional modifications to classes of deserialized or deserialized objects.For the CSVconverter we did not use any specific CSV libraries. We decided to handle the conversion ourselves.In more detail:

   1.1. **AirportsToString(List¡Airport¿ airports)**. It is used for serialization and deserialization of airport parameters.

   1.2. **CarriersToString(List¡Carrier¿ carriers)**. It is used for serialization and deserialization of carrier parameters.

   1.3. **StatisticToString**. It is used for serialization and deserialization of all statistics parameters.

   1.4. **StatisticToFlightString**. It is used for serialization and deserialization of all statistics about flights of a carrier from/to a US airport for a given period.

   1.5. **StatisticToDelayTimeString**.It is used for serialization and deserialization of the number of minutes of delay per carrier attributed to carrier-specific reasons/all reasons, for a given period.

   In both of these files we created methods that can easily understood by their names and the connection they have to the database.

   Last but not least we have the **DataConverter** file. This file contains all the methods used in the two previously mentioned java files.

2. **Models**

   Again this section in our code contains three files. Each files represents the three main classes that are important to the project along with the ability to return their parameters. Those are **Airport, Carrier, Statistics**.In more detail:

   2.1. **Airport**. Contains a single public class for specifing and returning the two parameters(name,code) of the Airport.

   2.2. **Carrier**. Contains a single public class for specifing and returning the two parameters(name,code) of the Carrier.

   2.3. **Statistics**.Contains a single public class for specifing and returning the two parameters of the Statistics. This includes parameters for recording both the number of times of delay due to different reasons as well as the minutes of delay for each of those reasons.

3. **Rest**

   This is the driver application of our project. From here we initiate all the functionality of our code by using the spring framework and libraries. For our application to navigate through all the data and parameters we made use of URI and HTTP .In more detail:

3.1. **Application.java**. Is the main driver function of our project.

3.2. **MainRestController.java**.Here we map the database through the usage of HTTP methods to navigate the different data of our code. The HTTP navigation can be found on section 2.3 of the report.In order to achieve the desired result we made use of Spring framework libraries. **HttpHeaders** is a data structure representing HTTP request or response headers, mapping String header names to a list of String values. **HttpStatus**is used for enumeration of HTTP status codes.**GetMapping**is annotation for mapping HTTP GET requests onto specific handler methods. **PathVariable** make it possible to have annotation which indicates that a method parameter should be bound to a URI template variable. Last but not least **RequestParam** indicates that a method parameter should be bound to a web request parameter.

4. **Database**

Lastly this section of our code is used for connecting to the database, to create the appropriate table insert or delete data as well as get the data we require on demand. Of course to achieve the we make use of SQL and jdbc. THe tables created are compliant with the CSV tables we have recieved for this project. The database start by requesting the credentials of the user in order to start creating the appropriate tables. Once the proper credantials have been met, the tables are created and connected to the SQL database. Airport,Carrier,Statistics each having their own tables. Furthermore we are able to manipulate these tables, as we are able to add or delete entries.