

# User Guide

ORFAP

Organisation for all purposes

## Getting Started

The **Flight-Analyzer**<sup>TM</sup> is a normal Web-Application, like `google.com` or `nytimes.com`. So to access it you need any one of these modern Webrowsers:

**Google Chrome** <https://www.google.de/chrome/browser/desktop/>

**Firefox** <https://www.mozilla.org/de/firefox/new/>

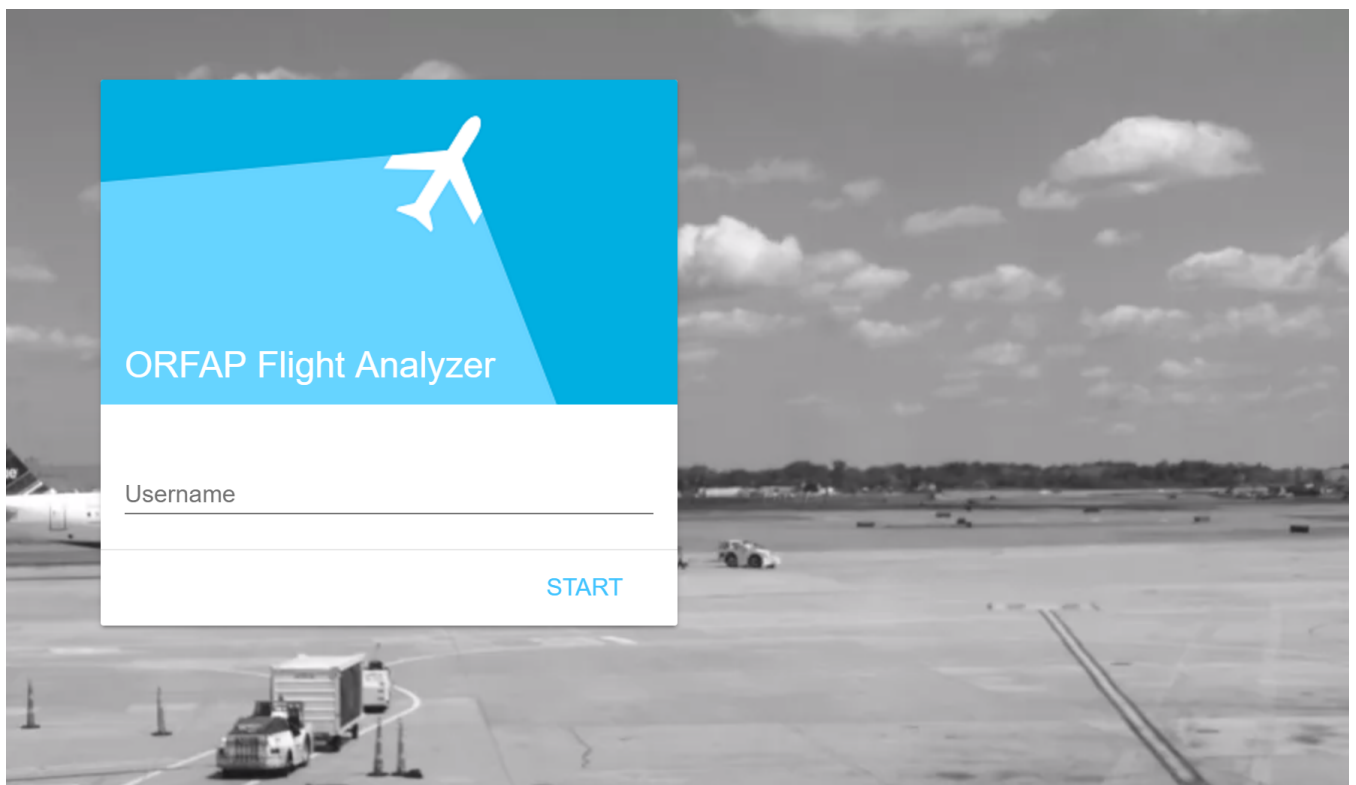
**Microsoft Edge** <https://www.microsoft.com/de-de/windows/microsoft-edge>

If you already have one of them installed you're good to go. If not you can follow these links to learn more about them and install one.

---

Once you have your browser up and running and are connected to the company network <sup>1</sup>, you can visit the **Flight-Analyzer**<sup>TM</sup> Application at `http://10.28.2.166/`.

## Logging in



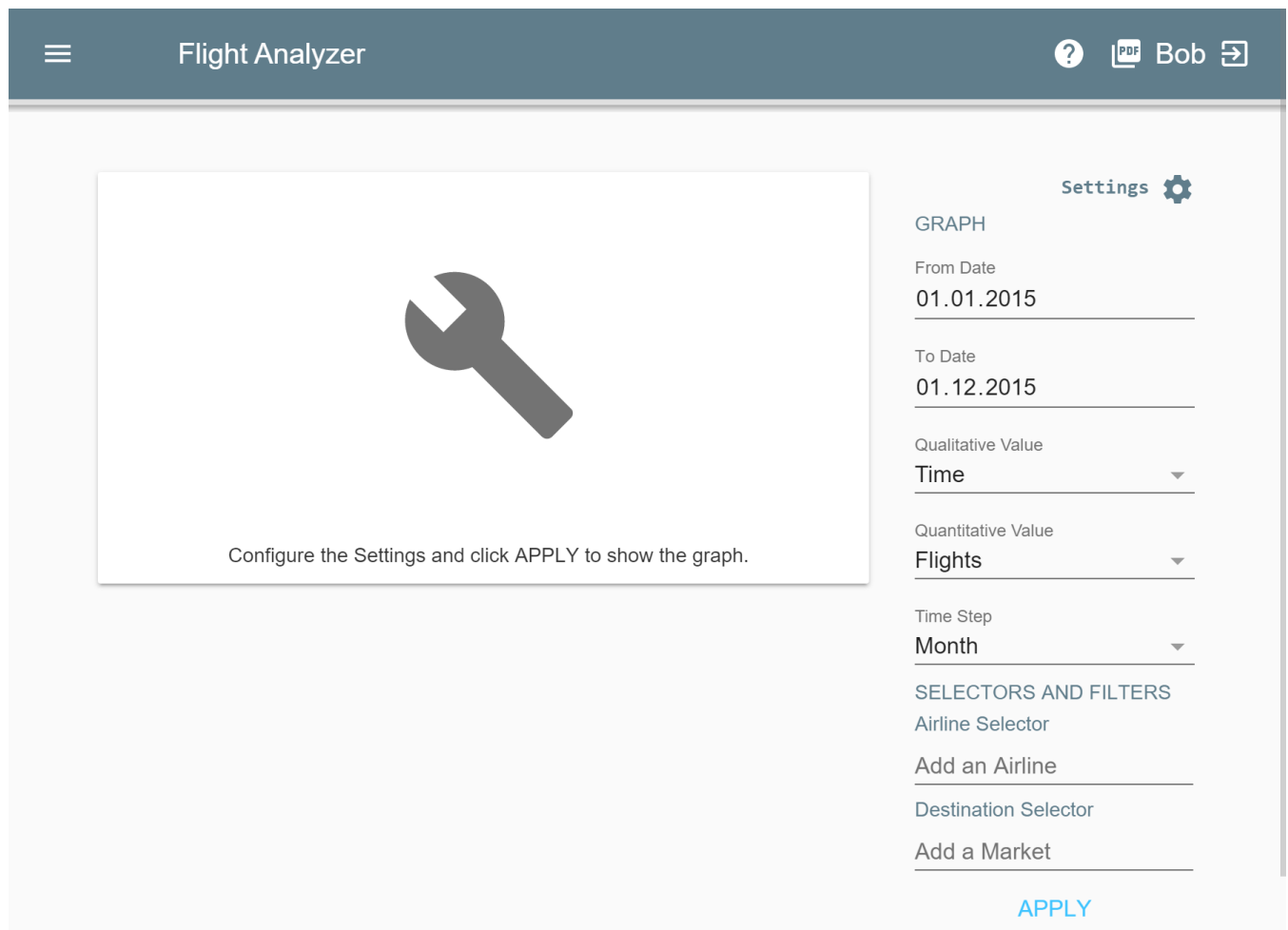
You are now presented with a splash screen. Here you are asked to insert your Username and click **START**. The username will be used to recognize you and store your personal settings.

---

<sup>1</sup>Currently, this means you are connected via VPN to the `1rz.de` network of the University of Applied Sciences Munich.

# The Pages and Navigation

## Graph Page

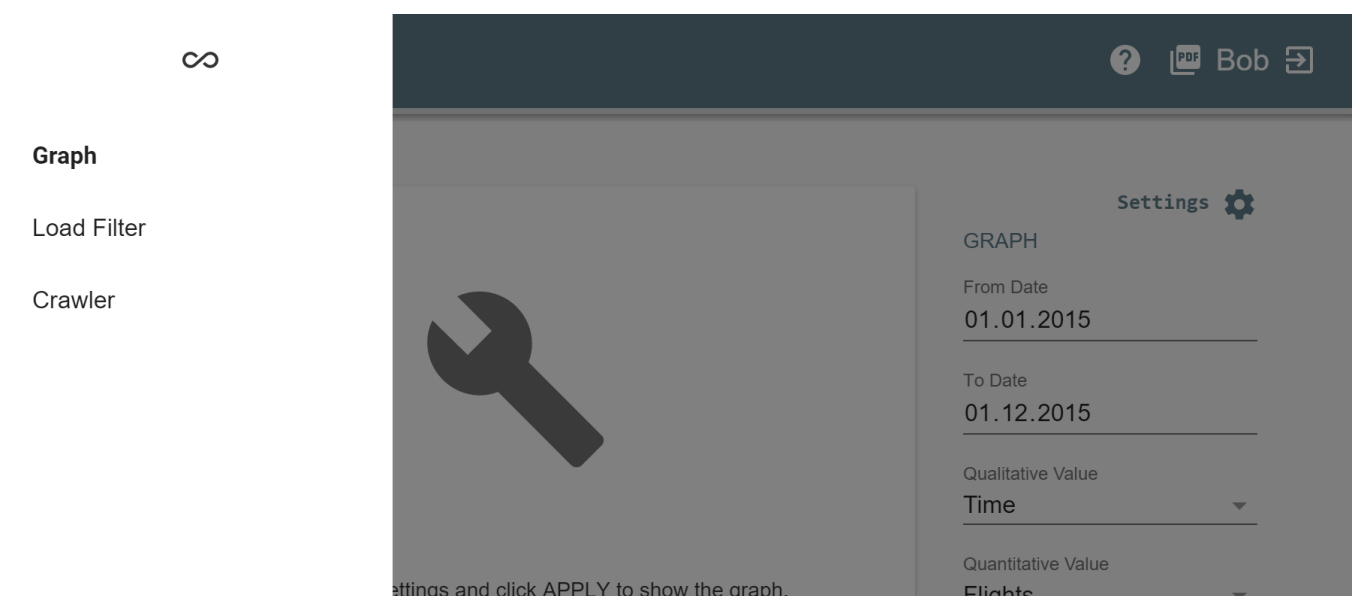


The screenshot shows the 'Flight Analyzer' application interface. At the top, a dark blue header bar contains a hamburger menu icon, the text 'Flight Analyzer', a help icon, a PDF icon, and the user name 'Bob' with a logout icon. The main content area is light grey. On the left, a large white box with a grey wrench icon contains the text 'Configure the Settings and click APPLY to show the graph.' On the right, a 'Settings' panel with a gear icon is visible. It includes sections for 'GRAPH' (From Date: 01.01.2015, To Date: 01.12.2015, Qualitative Value: Time, Quantitative Value: Flights, Time Step: Month) and 'SELECTORS AND FILTERS' (Airline Selector, Add an Airline, Destination Selector, Add a Market). A blue 'APPLY' button is at the bottom right of the settings panel.

Upon first login, you will see the **Graph Page**. This is the main page of the application and allows you to configure the settings for the graph and will show it once you click **APPLY**.

The configuration of the graph will be explained in an additional chapter.

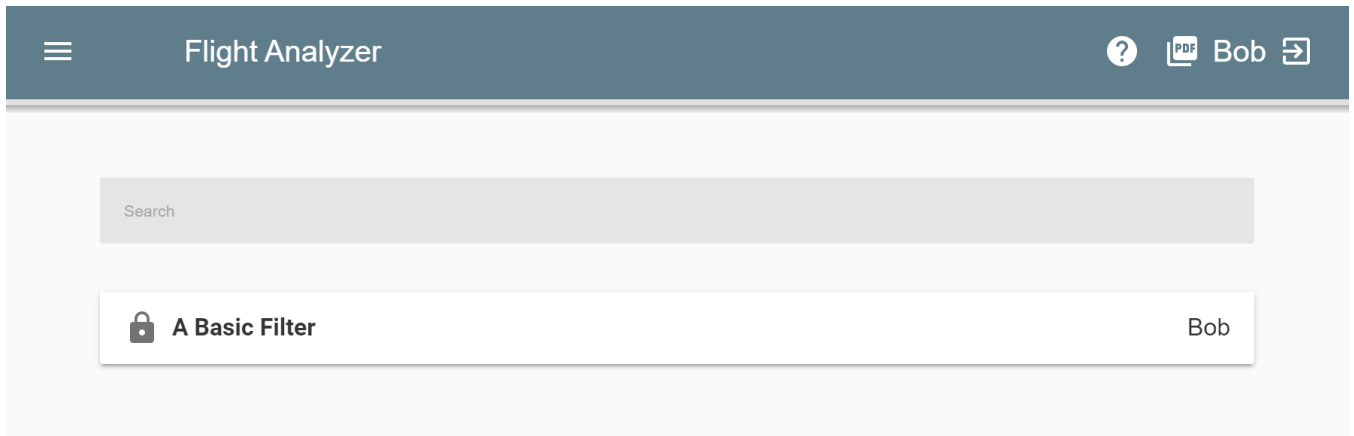
## Navigation



A click on the top left menu (*hamburger*) button will reveal the menu for changing pages. If you decide to not change the page you can still click on the large grey area and discard this action.

## Settings Page

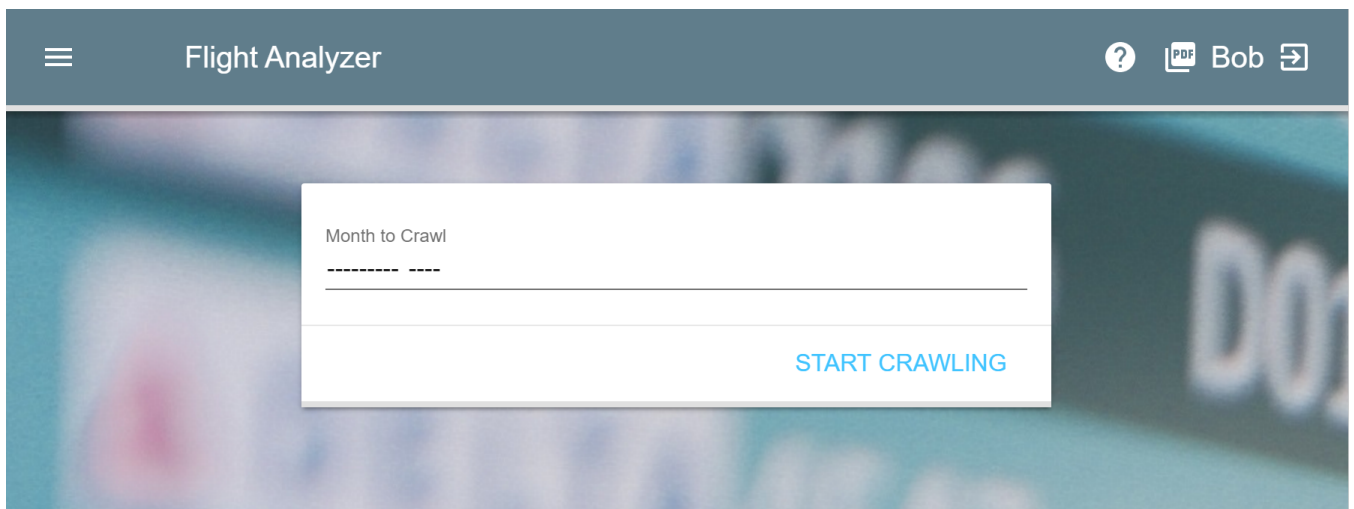
By selecting **Load Filter** in the Menu, you will see this page. Here you can load any stored filters/settings by selecting them and confirming the action with the button that appears in the bottom right corner.



The screenshot shows the 'Flight Analyzer' settings page. At the top is a dark blue header with a hamburger menu icon, the title 'Flight Analyzer', and user information including a question mark icon, a PDF icon, the name 'Bob', and an external link icon. Below the header is a light grey search bar with the placeholder text 'Search'. Underneath the search bar is a white card with a lock icon, the text 'A Basic Filter', and the name 'Bob' on the right side.

You can also limit the filters by typing into the light grey search field.

## Crawler Page




The screenshot shows the 'Flight Analyzer' crawler page. It features the same dark blue header as the settings page. The main content area has a blurred background image of a stadium. Overlaid on this is a white form with the label 'Month to Crawl' above a text input field. Below the input field is a blue button labeled 'START CRAWLING'.

Here you can start the Crawler to update the data from the **transtats.com** website. In the input field you have to specify the month to update. To start the crawling process click on **START CRAWLING**.

# Graph Configuration

## Basic Configuration

**Settings** 

**GRAPH**

From Date

01.01.2015

To Date

01.12.2015

Qualitative Value

Time

Quantitative Value

Flights

Time Step

Month

In here you can change the most fundamental way how and which data is displayed in the graph.

In the first two fields you select the time range to limit the data.

With the **Qualitative Value** you can select how the data is grouped. For example by selecting Destination, for each date (or date range) on the X-Axis multiple Bars (for all the different Destinations) will appear.

The **Quantitative Value** simply states what value should be used for the Y-Axis.

The **Time Step** picker sets the length of the period you are interested in. Adjust this Setting if you are interested in accumulated data for a whole year or the specific days of week.

## Selectors and Saving a Setting

SELECTORS AND FILTERS

Airline Selector

Add an Airline

Destination Selector

Add a Market

APPLY

Save Setting

☐ Public

Settings Name

SAVE

With this Selectors you can additionally limit data by Airline or Destination.

By adding an Airline to the Airline Selector, you tell the application that you are only interested for data of this Airline. You can also set a group (combination) of multiple Airlines. Choosing no Airline results in a collection of data for all Airlines.

The Destination Selector behaves exactly like the Airline Selector. By selecting multiple (or one) Markets the data will be limited to the flights that have these Destinations.

---

The **APPLY** button will calculate the data for the current filter settings and display them in the graph.

---

With the **Save Setting** Dialog you can store the current setting (filter) for only your personal use or shared for the whole company (public).

To store the Setting publically you need to check the **Public** box.

In the **Settings Name** field you can choose a fitting name for your Filter.

## Export as PDF

To export the Graph as a PDF simply click the PDF button on the top right in the toolbar. Then you can choose the target folder and name of the PDF and click Save to finish this process.

# Technical Documentation

ORFAP

Organisation for all purposes

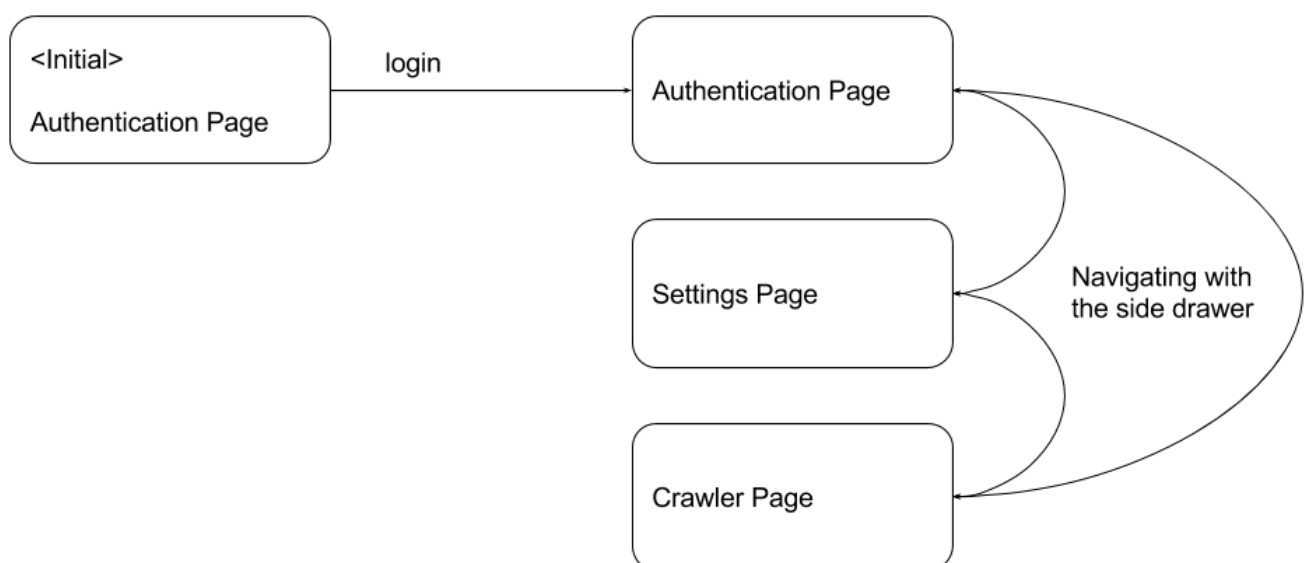
## Inhaltsverzeichnis

GUI . . . . .	1
Backend . . . . .	3
API Overview . . . . .	4
Airline . . . . .	4
Market . . . . .	5
Route . . . . .	7
Settings . . . . .	13
Crawler . . . . .	18
Format of the downloaded Tables from transtats . . . . .	18
Flowchart for the crawling process . . . . .	21
Definitions . . . . .	22
Crawler API . . . . .	22

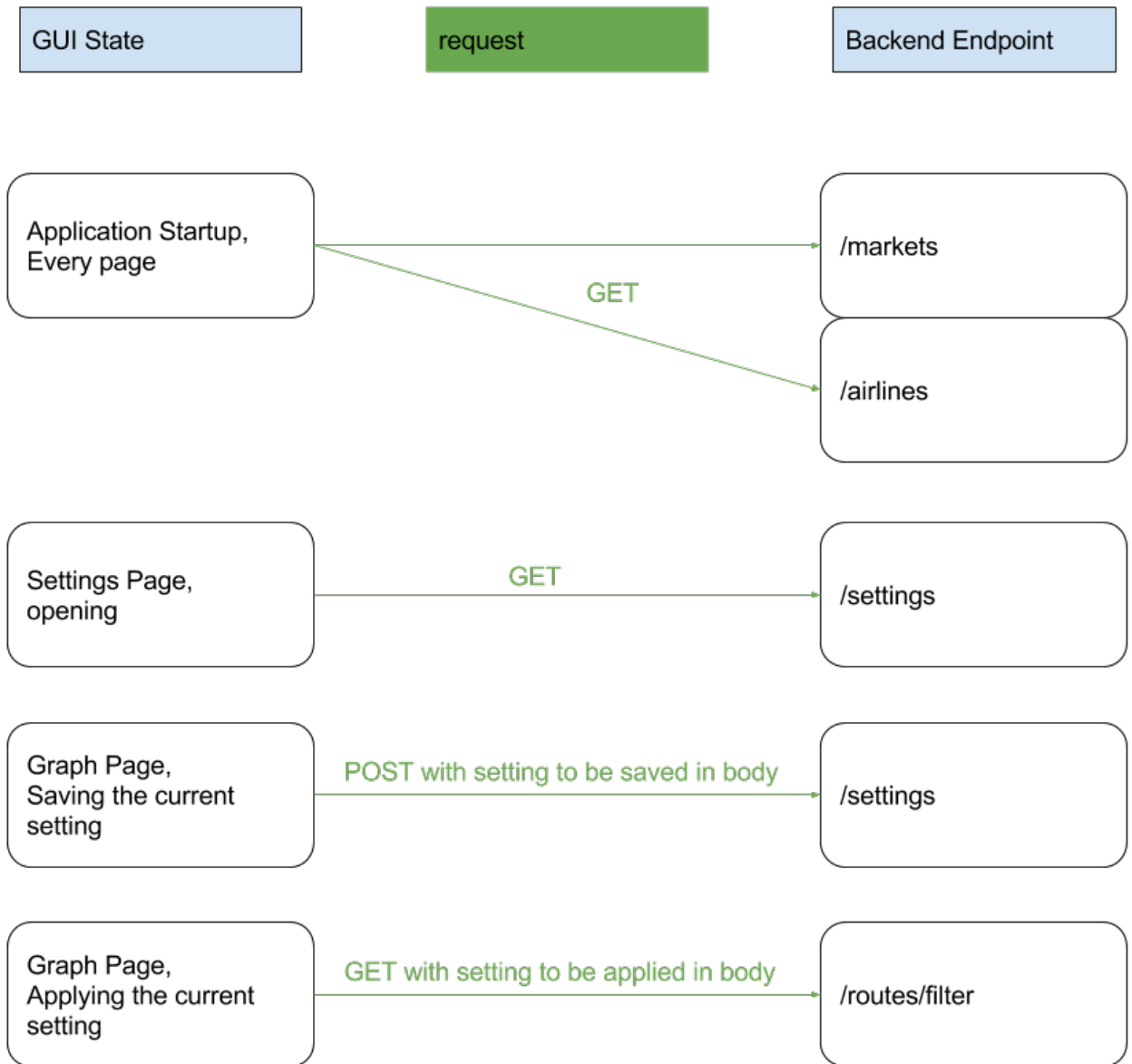
## GUI

The GUI offers a visual representation for the backends API which is described further down in this document. The only data which is stored locally is the last entered username of the user. This username is stored inside a cookie called **username** but is not needed for the correct functionality of the application.

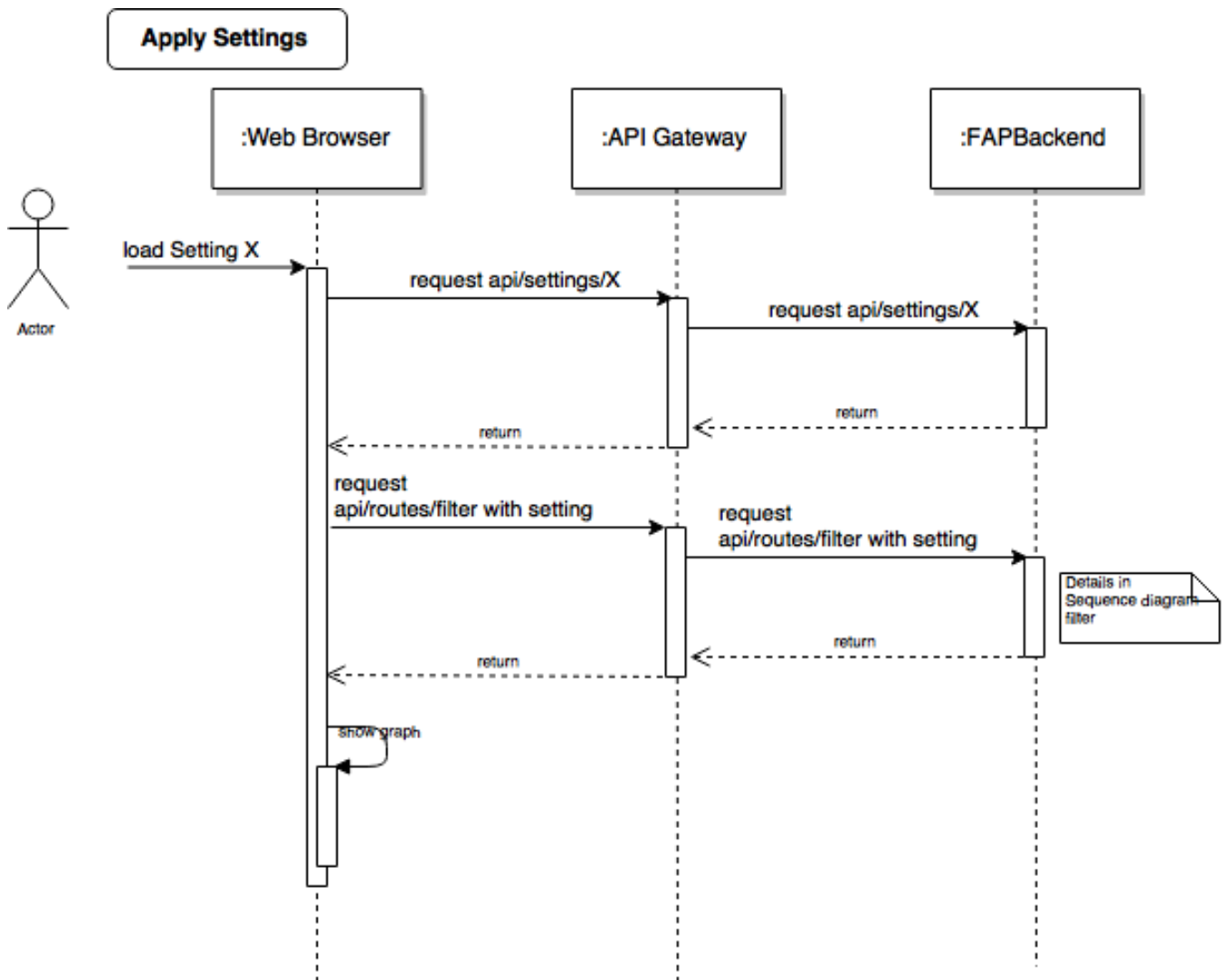
The application can be in one of four different states during use:



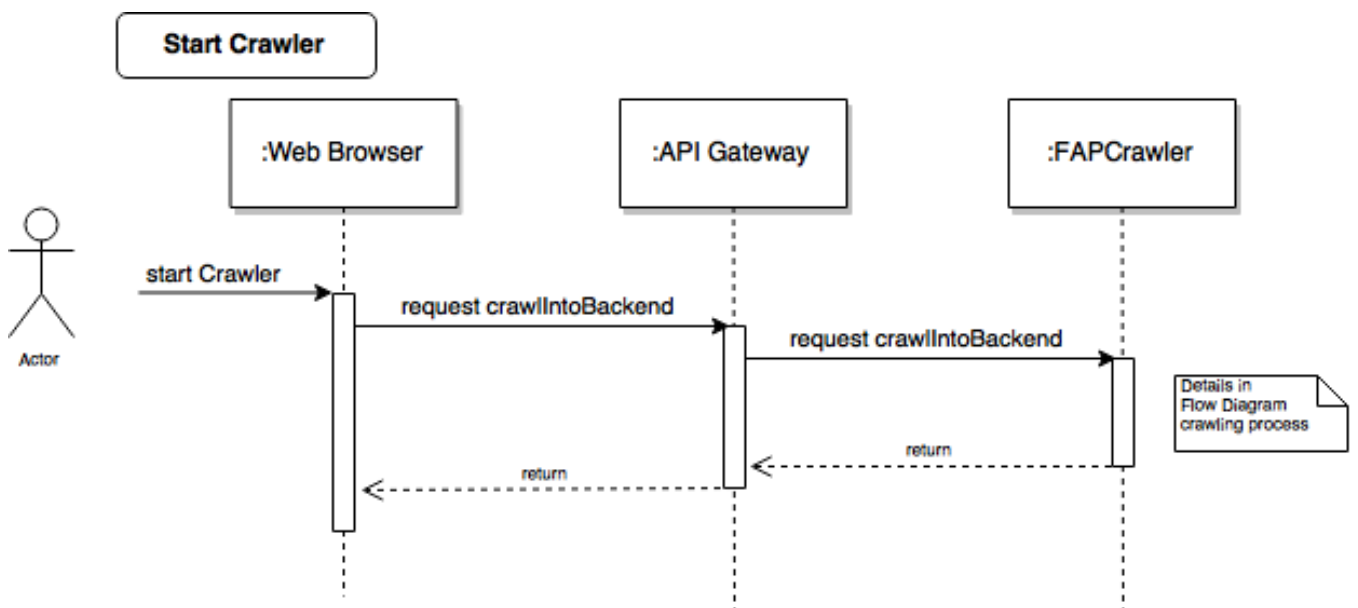
The following flowchart displays the requests from the frontend in the different states. A refresh in the browser restarts these requests.



The next sequence diagram shows the background workflow of loading and automatically applying a setting.



With the GUI the user can start the crawler to fetch the latest data from transtats. This is shown in the next sequence diagram.



## Backend

The Backend holds the whole data and offers an API for it. The following section will describe this API. For every Entity on the database the necessary endpoints (Path) will be shown with the depending HTTP Methods to call.



Base URL: <http://10.28.2.166/api>

## API Overview

Entity	Path	HTTP Methods
Airline	/airlines	GET, POST
Airline	/airlines/{id}	GET, PUT, DELETE
Market	/markets	GET, POST
Market	/markets/{id}	GET, PUT, DELETE
Route	/routes	GET, POST
Route	/routes/{id}	GET, PUT, DELETE
Route	/routes/saveAll	POST
Route	/routes/search/isRouteInMonth	GET
Route	/routes/filter	GET

## Airline

### Entity Schema

Property	Type	Required
id	String	yes
name	String	yes

### Path /airlines

#### Http Methods

##### Get

**Description:** Get all saved airlines.

##### Response Example:

Code 200

```
[
  {
    "id": "213123",
    "name": "Lufthansa"
  },
  {
    "id": "12312",
    "name": "AirBerlin"
  }
]
```

##### Post

**Description:** Save an airline.

##### Request Example:

Header: Content-Type application/json

```
{
  "id": "213123",
  "name": "Lufthansa"
}
```

```
}
```

### Response Example:

Code 201

```
{
  "id": "213123",
  "name": "Lufthansa"
}
```

**Path** /airlines/{id}

### Http Methods

#### *Get*

**Description:** Get an specific airline with the given id.

### Response Example:

Code 200

```
{
  "id": "213123",
  "name": "Lufthansa"
}
```

#### *Put*

**Description:** Update an specific airline with the given id.

### Request Example:

Header: Content-Type application/json

```
{
  "id": "213123",
  "name": "Lufthansa"
}
```

### Response Example:

Code 200

```
{
  "id": "213123",
  "name": "Lufthansa"
}
```

#### *Delete*

**Description:** Deletes an specific airline with the given id.

### Response Example:

Code 204

### Market

### Entity Schema

Property	Type	Required
id	String	yes

Property	Type	Required
name	String	yes

## **Path** /markets

### **Http Methods**

#### *Get*

**Description:** Get all saved markets.

#### **Response Example:**

Code 200

```
[
  {
    "id": "213123",
    "name": "New York"
  },
  {
    "id": "12312",
    "name": "Colorado"
  }
]
```

#### *Post*

**Description:** Save a market.

#### **Request Example:**

Header: Content-Type application/json

```
{
  "id": "213123",
  "name": "New York"
}
```

#### **Response Example:**

Code 201

```
{
  "id": "213123",
  "name": "New York"
}
```

## **Path** /markets/{id}

### **Http Methods**

#### *Get*

**Description:** Get a specific market with the given id.

#### **Response Example:**

Code 200

```
{
  "id": "213123",
  "name": "New York"
}
```

#### *Put*

**Description:** Update a specific market with the given id.

#### **Request Example:**

Header: Content-Type application/json

```
{
  "id": "213123",
  "name": "New York"
}
```

#### **Response Example:**

Code 200

```
{
  "id": "213123",
  "name": "New York"
}
```

#### *Delete*

**Description:** Deletes a specific market with the given id.

#### **Response Example:**

Code 204

### **Route**

#### **Entity Schema**

Property	Type	Required
id	String	no
date	Date/String	yes
delays	double	no
cancelled	double	no
passengerCount	double	no
flightCount	double	no
airline	Link	yes
source	Link	yes
destination	Link	yes

#### **Path /routes**

##### **Http Methods**

#### *Get*

**Description:** Get all saved routes.

#### **Response Example:**

Code 200

```
[
  {
    "date": "2015-12-01",
    "delays": 10,
    "cancelled": 0,
    "passengerCount": 130,
    "flightCount": 1,
    "airline": "http://10.28.2.166/api/airlines/123123",
    "source": "http://10.28.2.166/api/markets/23423424",
    "destination": "http://10.28.2.166/api/markets/1231231"
  },
  {
    "date": "2015-10-20",
    "delays": 15,
    "cancelled": 0,
    "passengerCount": 120,
    "flightCount": 1,
    "airline": "http://10.28.2.166/api/airlines/123123",
    "source": "http://10.28.2.166/api/markets/23423424",
    "destination": "http://10.28.2.166/api/markets/1231231"
  }
]
```

#### **Post**

**Description:** Save a route.

#### **Request Example:**

Header: Content-Type application/json

```
{
  "date": "2015-12-01",
  "delays": 10,
  "cancelled": 0,
  "passengerCount": 130,
  "flightCount": 1,
  "airline": "http://10.28.2.166/api/airlines/123123",
  "source": "http://10.28.2.166/api/markets/23423424",
  "destination": "http://10.28.2.166/api/markets/1231231"
}
```

#### **Response Example:**

Code 201

```
{
  "date": "2015-12-01",
  "delays": 10,
  "cancelled": 0,
  "passengerCount": 130,
  "flightCount": 1,
  "airline": "http://10.28.2.166/api/airlines/123123",
  "source": "http://10.28.2.166/api/markets/23423424",
  "destination": "http://10.28.2.166/api/markets/1231231"
}
```

**Path** /routes/saveAll

## Http Methods

### Post

**Description:** Saves a list of routes.

### Request Example:

Header: Content-Type application/json

```
[
  {
    "date": "2015-12-01",
    "delays": 10,
    "cancelled": 0,
    "passengerCount": 130,
    "flightCount": 1,
    "airline": "123123",
    "source": "23423424",
    "destination": "1231231"
  },
  {
    "date": "2015-10-20",
    "delays": 15,
    "cancelled": 0,
    "passengerCount": 120,
    "flightCount": 1,
    "airline": "123123",
    "source": "23423424",
    "destination": "1231231"
  }
]
```

### Response Example:

Code 200

**Path** /routes/search/isRouteInMonthOfYear

## Http Methods

### Get

**Description:** Determine if there are already routes saved for the given month of year.

### Query

- **date** Date to check with format: yyyy-MM

### Response Example:

Code 200

true

**Path** /routes/{id}

## Http Methods

### Get

**Description:** Get a specific route with the given id.

### Response Example:

Code 200

```
{
  "date": "2015-12-01",
  "delays": 10,
  "cancelled": 0,
  "passengerCount": 130,
  "flightCount": 1,
  "airline": "http://10.28.2.166/api/airlines/123123",
  "source": "http://10.28.2.166/api/markets/23423424",
  "destination": "http://10.28.2.166/api/markets/1231231"
}
```

### Put

**Description:** Update a specific route with the given id.

### Request Example:

Header: Content-Type application/json

```
{
  "date": "2015-12-01",
  "delays": 10,
  "cancelled": 0,
  "passengerCount": 130,
  "flightCount": 1,
  "airline": "http://10.28.2.166/api/airlines/123123",
  "source": "http://10.28.2.166/api/markets/23423424",
  "destination": "http://10.28.2.166/api/markets/1231231"
}
```

### Response Example:

Code 200

```
{
  "date": "2015-12-01",
  "delays": 10,
  "cancelled": 0,
  "passengerCount": 130,
  "flightCount": 1,
  "airline": "http://10.28.2.166/api/airlines/123123",
  "source": "http://10.28.2.166/api/markets/23423424",
  "destination": "http://10.28.2.166/api/markets/1231231"
}
```

### Delete

**Description:** Deletes a specific route with the given id.

### Response Example:

Code 204

**Path** /routes/filter

### Http Methods

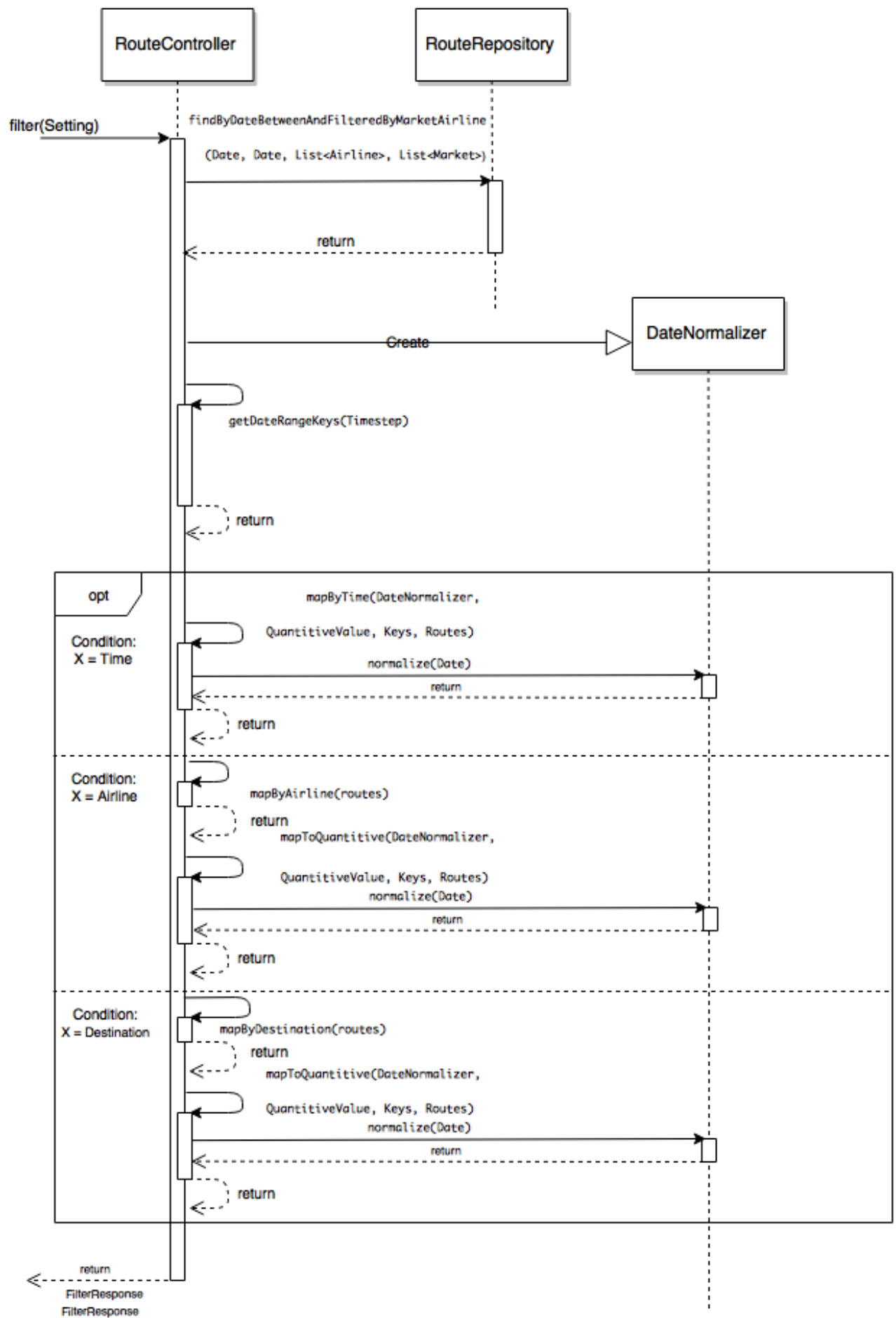
#### Get

**Description:** Apply a given filter setting to the database and provide preformatted data.

This action is not a simple database request and includes business logic. The following sequence diagram describes this logic:



# Filter data on given settings



Request Example:

Header: Content-Type application/json

```
{
  "name": "Test-View",
  "creator": "Hans",
  "shareable": true,
  "rangeFrom": "2015-01-01",
  "rangeTo": "2015-03-31",
  "filter": {
    "destinations": [
      "23123123","2313123123"
    ],
    "airlines": [
      "678678","6786786867"
    ],
    "timestep": "MONTH"
  },
  "axis": {
    "x": "TIME",
    "y": "FLIGHTS"
  }
}
```

### Response Example:

Code 200

```
{
  "y": "FLIGHTS",
  "z": "TIME",
  "x": "January, February, March",
  "data":{
    "January": 20,
    "February": 15,
    "March": 21
  }
}
```

### Settings

#### Entity Schema

Property	Type	Required
id	String	no
name	String	yes
creator	String	yes
shareable	boolean	no
rangeFrom	Date/String	no
rangeTo	Date/String	no
filter	Filter	yes
axis	Axis	yes

#### Path /settings

#### Http Methods

## Get

**Description:** Get all saved settings.

## Response Example:

Code 200

```
[
  {
    "name": "Standard-View",
    "creator": "Hans",
    "shareable": true,
    "rangeFrom": "2015-01-01",
    "rangeTo": "2015-12-31",
    "filter": {
      "destinations": [
        "23123123", "2313123123"
      ],
      "airlines": [
        "678678", "6786786867"
      ],
      "timestep": "MONTH"
    },
    "axis": {
      "x": "TIME",
      "y": "FLIGHTS"
    }
  },
  {
    "name": "Extended-View",
    "creator": "Hans",
    "shareable": true,
    "rangeFrom": "2015-01-01",
    "rangeTo": "2015-01-31",
    "filter": {
      "destinations": [
        "23123123", "2313123123"
      ],
      "airlines": [
        "678678", "6786786867"
      ],
      "timestep": "DAY_OF_WEEK"
    },
    "axis": {
      "x": "DESTINATION",
      "y": "PASSENGERS"
    }
  }
]
```

## Post

**Description:** Save a setting.

## Request Example:

Header: Content-Type application/json

```
{
  "name": "Standard-View",
```

```
"creator": "Hans",
"shareable": true,
"rangeFrom": "2015-01-01",
"rangeTo": "2015-12-31",
"filter": {
  "destinations": [
    "23123123","2313123123"
  ],
  "airlines": [
    "678678","6786786867"
  ],
  "timestep": "MONTH"
},
"axis": {
  "x": "TIME",
  "y": "FLIGHTS"
}
}
```

### Response Example:

Code 201

```
{
  "name": "Standard-View",
  "creator": "Hans",
  "shareable": true,
  "rangeFrom": "2015-01-01",
  "rangeTo": "2015-12-31",
  "filter": {
    "destinations": [
      "23123123","2313123123"
    ],
    "airlines": [
      "678678","6786786867"
    ],
    "timestep": "MONTH"
  },
  "axis": {
    "x": "TIME",
    "y": "FLIGHTS"
  }
}
```

**Path** /settings/{id}

### Http Methods

*Get*

**Description:** Get a specific setting with the given id.

### Response Example:

Code 200

```
{
  "name": "Standard-View",
  "creator": "Hans",
```

```
"shareable": true,
"rangeFrom": "2015-01-01",
"rangeTo": "2015-12-31",
"filter": {
  "destinations": [
    "23123123","2313123123"
  ],
  "airlines": [
    "678678","6786786867"
  ],
  "timestep": "MONTH"
},
"axis": {
  "x": "TIME",
  "y": "FLIGHTS"
}
}
```

**Put**

**Description:** Update a specific setting with the given id.

**Request Example:**

Header: Content-Type application/json

```
{
  "name": "Standard-View",
  "creator": "Hans",
  "shareable": true,
  "rangeFrom": "2015-01-01",
  "rangeTo": "2015-12-31",
  "filter": {
    "destinations": [
      "23123123","2313123123"
    ],
    "airlines": [
      "678678","6786786867"
    ],
    "timestep": "MONTH"
  },
  "axis": {
    "x": "TIME",
    "y": "FLIGHTS"
  }
}
```

**Response Example:**

Code 200

```
{
  "name": "Standard-View",
  "creator": "Hans",
  "shareable": true,
  "rangeFrom": "2015-01-01",
  "rangeTo": "2015-12-31",
  "filter": {
    "destinations": [
      "23123123","2313123123"
    ],

```

```

    "airlines": [
      "678678","6786786867"
    ],
    "timestep": "MONTH"
  },
  "axis": {
    "x": "TIME",
    "y": "FLIGHTS"
  }
}

```

#### **Delete**

**Description:** Deletes a specific setting with the given id.

#### **Response Example:**

Code 204

**Path** /settings/search/ findByNameContainingIgnoreCaseOrCreatorContainingIgnoreCase

### **Http Methods**

#### **Get**

**Description:** Find settings by name or by creator name.

#### **Query**

- **name** Name of setting
- **creator** Name of creator

#### **Response Example:**

Code 200

```

[
  {
    "name": "Standard-View",
    "creator": "Hans",
    "shareable": true,
    "rangeFrom": "2015-01-01",
    "rangeTo": "2015-12-31",
    "filter": {
      "destinations": [
        "23123123","2313123123"
      ],
      "airlines": [
        "678678","6786786867"
      ],
      "timestep": "MONTH"
    },
    "axis": {
      "x": "TIME",
      "y": "FLIGHTS"
    }
  }
]

```

**Path** /settings/search/ findByCreatorContainingIgnoreCaseOrShareableTrue

## Http Methods

### Get

**Description:** Find settings by creator name including all public settings.

### Query

- **creator** Name of creator

### Response Example:

Code 200

```
[
  {
    "name": "Standard-View",
    "creator": "Hans",
    "shareable": true,
    "rangeFrom": "2015-01-01",
    "rangeTo": "2015-12-31",
    "filter": {
      "destinations": [
        "23123123", "2313123123"
      ],
      "airlines": [
        "678678", "6786786867"
      ],
      "timestep": "MONTH"
    },
    "axis": {
      "x": "TIME",
      "y": "FLIGHTS"
    }
  }
]
```

## Crawler

### Format of the downloaded Tables from transtats

#### Airline Lookup table

- AirlineID [http://transtats.bts.gov/Download\\_Lookup.asp?Lookup=L\\_AIRLINE\\_ID](http://transtats.bts.gov/Download_Lookup.asp?Lookup=L_AIRLINE_ID)

#### Example

```
http.Get("http://transtats.bts.gov/Download_Lookup.asp?Lookup=L_AIRLINE_ID")
```

Code,Description

```
"19031","Mackey International Inc.: MAC"
"19032","Munz Northern Airlines Inc.: XY"
"19033","Cochise Airlines Inc.: COC"
"19034","Golden Gate Airlines Inc.: GSA"
"19035","Aeromech Inc.: RZZ"
"19036","Golden West Airlines Co.: GLW"
"19037","Puerto Rico Intl Airlines: PRN"
"19038","Air America Inc.: STZ"
"19039","Swift Aire Lines Inc.: SWT"
```

```
{
  "name": "string[[Description]]",
  "id": "int[[Code]]"
}
```

## Market Lookup table

- Dest/OriginCityMarketID [http://transtats.bts.gov/Download\\_Lookup.asp?Lookup=L\\_CITY\\_MARKET\\_ID](http://transtats.bts.gov/Download_Lookup.asp?Lookup=L_CITY_MARKET_ID)

### Example

```
http.Get("http://transtats.bts.gov/Download_Lookup.asp?Lookup=L_CITY_MARKET_ID")
```

```
Code,Description
"30001","Afognak Lake, AK"
"30003","Granite Mountain, AK"
"30004","Lik, AK"
"30005","Little Squaw, AK"
"30006","Kizhuyak, AK"
"30007","Klawock, AK"
"30008","Elizabeth Island, AK"
"30009","Homer, AK"
"30010","Hudson, NY"
```

```
{
  "name": "string[[Description]]",
  "id": "int[[Code]]"
}
```

## Route table

- T-100 Domestic Segment (All Carriers)  
[http://transtats.bts.gov/DL\\_SelectFields.asp?Table\\_ID=311](http://transtats.bts.gov/DL_SelectFields.asp?Table_ID=311)

```
"YEAR", "MONTH", "DEPARTURES_SCHEDULED", "DEPARTURES_PERFORMED", "PASSENGERS", "AIRLINE_ID", "ORIGIN
```

### Example

```
form := url.Values{"sqlstr": {"SELECT YEAR,MONTH,DEPARTURES_SCHEDULED,DEPARTURES_PERFORMED,PAS
                                ORIGIN_CITY_MARKET_ID,DEST_CITY_MARKET_ID,MONTH
                                FROM T_T100D_SEGMENT_ALL_CARRIER
                                AND YEAR=[[year]]
                                AND MONTH=[[month]]
                                AND ORIGIN_CITY_MARKET_ID=31703"}}}
```

```
http.PostForm("http://transtats.bts.gov/DownLoad_Table.asp", form)
```

```
"DEPARTURES_SCHEDULED", "DEPARTURES_PERFORMED", "PASSENGERS", "AIRLINE_ID", "ORIGIN_CITY_MARKET_ID"
0.00,1.00,0.00,21107,31703,31703,2,
0.00,1.00,0.00,21492,31703,31995,2,
0.00,1.00,0.00,21492,31703,31703,2
```

```
{
  "date": "string[[YEAR]]-[[MONTH]]-01",
  "delays": "0",
  "cancelled": "0",
  "passengerCount": "double[[PASSENGERS]]",
  "flightCount": "0",
  "airline": "hal+id[[AIRLINE_ID]]",
  "source": "hal+id[[ORIGIN_CITY_MARKET_ID]]",
}
```



```

    "destination": "hal+id[[DEST_CITY_MARKET_ID]]"
}

```

## Flights table

- On-Time Performance [http://transtats.bts.gov/DL\\_SelectFields.asp?Table\\_ID=236](http://transtats.bts.gov/DL_SelectFields.asp?Table_ID=236)

```

"DAY_OF_WEEK", "FL_DATE", "AIRLINE_ID", "ORIGIN_CITY_MARKET_ID", "DEST_CITY_MARKET_ID", "ARR_DELAY_

```

## Example

```

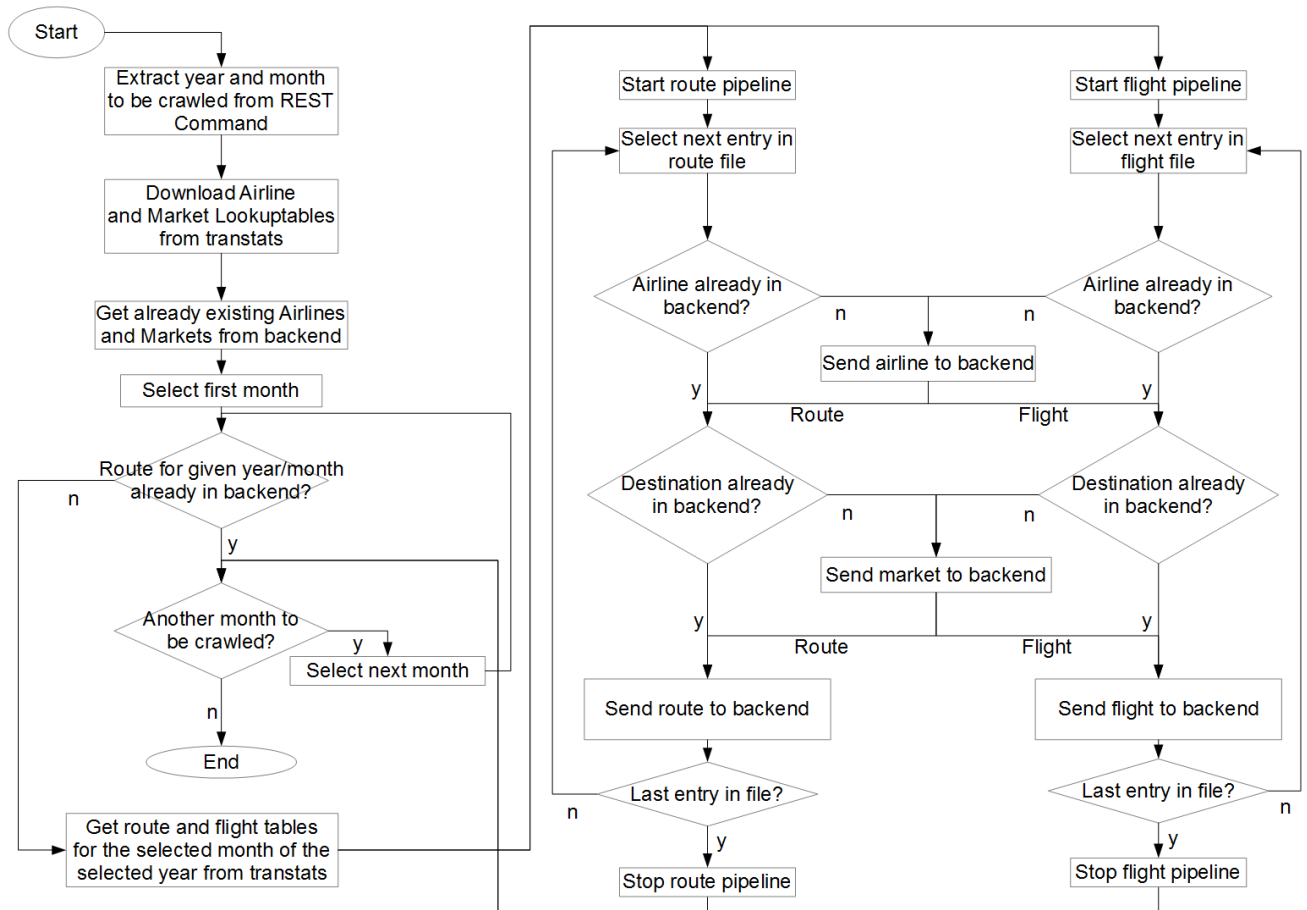
form := url.Values{"sqlstr": {"SELECT DAY_OF_WEEK,FL_DATE,AIRLINE_ID,ORIGIN_CITY_MARKET_ID,DES
                                ARR_DELAY_NEW,CANCELLED
                                FROM T_ONTIME
                                WHERE Month=[[month]]
                                AND YEAR=[[year]]
                                AND ORIGIN_CITY_MARKET_ID=31703"}}
http.PostForm("http://transtats.bts.gov/DownLoad_Table.asp", form)

"DAY_OF_WEEK", "FL_DATE", "AIRLINE_ID", "ORIGIN_CITY_MARKET_ID", "DEST_CITY_MARKET_ID", "ARR_DELAY_
1,2015-02-02,19805,31703,32575,,1.00,
1,2015-02-09,19805,31703,32575,22.00,0.00,
1,2015-02-16,19805,31703,32575,0.00,0.00,
1,2015-02-23,19805,31703,32575,9.00,0.00

{
    "date": "string[[FL_DATE]]",
    "delays": "double[[ARR_DELAY_NEW]]",
    "cancelled": "double[[CANCELLED]]",
    "passengerCount": "0",
    "flightCount": "1",
    "airline": "hal+id[[AIRLINE_ID]]",
    "source": "hal+id[[ORIGIN_CITY_MARKET_ID]]",
    "destination": "hal+id[[DEST_CITY_MARKET_ID]]"
}

```

## Flowchart for the crawling process



## Detailed listing of the proceedings of the crawling process

1. Start the crawler via a REST command, either by using the “Crawler”-Button in the GUI or by just sending the command to <http://10.28.2.166/crawler>.  
The command has the following syntax: <http://<serverIPAddress>/crawler/crawlIntoBackend?year=<year>> (8)  
The year must be a number, so must the month. If just the year is provided, the full year will be crawled.  
Months can be given as discrete months or in ranges, e.g. 1-7.  
The part in () brackets is optional.
2. The crawler now starts the crawling process. This process is divided in several steps.
3. The list of all airlines available is downloaded from the transtats server and they are saved in a hashmap. Also all airlines already in the database are requested and saved into a set.
4. The list of all markets available is downloaded from the transtats server and they are saved in a hashmap. Also all markets already in the database are requested and saved into a set.
5. The given year and month or months, are extracted and a loop is created, starting with the first given month, ending with the last given month.
6. The T100D database is queried for all routes originating in the NYC market, going to other markets in the US, in the first given month.
7. The route pipeline is created.
8. The zip-file containing the routes is extracted entry by entry and pushed into the route pipeline.
9. Routes are created from the given entry.
10. Invalid routes are filtered.
11. Airlines and markets used in the created route and not already in the database are written into the database.
12. The route is written into the database
13. Step 6 to 12 are done for the on-time-database, now creating (logical) flights instead of routes.

14. If there is more than one month to be crawled, steps 6 to 13 are repeated for every given month.

## Definitions

- **Route:** Information available on a monthly base. They contain the number of passengers going from one market to the other in the given month. The information for number of passengers is always given as a route with the date of the first day of the corresponding month.
- **Flight:** Information available on a daily base. This contains, for each flight, information about delays and cancellations, as well as the exact date of the flight. Flights also make up the number of flights from one market to another.
- **Market:** A market includes all airports of a domestic region. E.g. the domestic region New York includes 13 airports.

## Crawler API

Base URL: `http://10.28.2.166/crawler`

**Path** `/crawlIntoBackend`

### Http Methods

*Get*

**Description:** Crawl new data into the backend.

**Query**

- **year** Year to crawl with format: yyyy
- **month** Range of months to crawl (optional) with the format: mm-mm or mm

**Response Example:**

Code 200

# Technical Installation Guide

ORFAP

Organisation for all purposes

## Server Side Installation

To deploy the application on your companies server you need to install docker on the server you want to deploy the application on. A docker installation guide can be found <https://docs.docker.com/linux/>.

Further you need to make sure you have a MYSQL-Database setup with the following data: \* Username: ExtDev2 \* Password: \*\*\*

To setup a MYSQL-Database on the server please refer to <http://dev.mysql.com/doc/refman/5.7/en/linux-ins>

Now you are ready to go to get the application up and running! From here it is as simple as executing some commands.

- Starting the Backend: `docker run -d -e TZ=GMT+2 -e "SPRING_PROFILES_ACTIVE=production" -p 8081:8080 darenegade/fapbackend`
- Starting the Crawler: `sudo docker run -d -p 8082:8081 arne2/fapcrawler`
- Starting the GUI: `sudo docker run -d -p 8080:80 petermueller/flight-analyzer`
- Starting the API-Gateway: `sudo docker run -d -p 80:8080 darenegade/fapapigateway`

If you want you can also start the so called Watchtower. This docker container will keep all parts of the applications updated automatically when a new version is released.

- Starting the watchtower: `sudo docker run -d -v /var/run/docker.sock:/var/run/docker.sock centurylink/watchtower`

Some more neat commands to get you started with using docker:

- `sudo docker ps` lists all running containers. The `-a` flag lists all stopped containers, too.
- `sudo docker logs <id>` shows the log output of this container. `-f` tails the log so you can continue watching the output.
- `sudo docker stop <id(s)>` stops all given containers.
- `sudo docker stats <id(s)>` show the current system usage of all given containers.

Now just wait for all containers to start and you can continue by crawling your first data. How to do this? Just follow the user documentation!

## Client Side Installation

To get the application running on your machine you only need two things.

1. You need a connection to your companies network. You can connect to your companies network with the companies VPN. You can find an instruction how to connect to it [https://www.lrz.de/services/netz/mobil/vpn\\_en/](https://www.lrz.de/services/netz/mobil/vpn_en/)
2. A modern web browser. It doesn't matter if it's Firefox, Chrome or Microsoft Edge. Only Internet Explorer is not supported.

Now you are ready to go! Just go to 10.28.2.166 and login with your name. For further instruction on how to use the application please refer to the user guide.