Air Flux Capacity

It is Friday, the 13th of April 2018, and your new job at Air Flux has been quite smooth so far. But this morning, as you come to the office for the early shift at 7am, everything is different: the old flight operations manager has decided to leave to Bora-Bora and left very little information behind how to actually operate an airline.

Everyone is desperate – there are flights on the schedule, but the airports have to know which planes to expect. Also, the flight captains have to get their plan for the day, which airport to fly to when.

There is only two pieces of information you have: 1. The home bases of your aircraft, conveniently it is one airplane per airport you're flying from and to, 2. The flight schedules as they were published to the airports.

The Information

The aircraft are based as follows:

```
Boeing 737 – Berlin, registration FL-0001
Airbus A321 – Munich, registration FL-0002
Boeing 747-400 – London, registration FL-0003
Airbus A320 - Hamburg, registration FL-0004
```

The flight schedules are as follows (time of departure, origin, destination, flighttime):

```
TXL
===================
10:00 TXL MUC 01:00
15:00 TXL MUC 01:00
16:00 TXL MUC 01:00
18:00 TXL MUC 01:00
21:00 TXL HAM 00:40

MUC
===================
10:00 MUC LHR 02:00
13:00 MUC TXL 01:00
15:00 MUC TXL 01:00
15:30 MUC LHR 02:00
17:30 MUC HAM 01:00
18:00 MUC LHR 02:30
20:00 MUC LHR 02:00
22:00 MUC TXL 01:00

LHR
===================
09:00 LHR HAM 02:30
12:00 LHR TXL 02:00
```

```
17:00 LHR TXL 02:00
20:30 LHR MUC 02:00

HAM
===================
10:00 HAM MUC 01:00
13:00 HAM MUC 01:00
20:00 HAM MUC 01:00
```

Your Task

Your task is to save the company, of course. With an API - you never thought this day would ever come!

After a quick discussion with the team, you decide to create an API with two endpoints

Help the airports assign gates that fit the departing aircraft

`GET /flightplan`, which returns a json list of Flight objects like this:

```
{
  "origin": "TXL",
  "destination": "MUC",
  "departure": "2018-04-13T10:00:00.000+02:00",
  "arrival": "2018-04-13T11:00:00.000+02:00",
  "equipment": "737"
}
```

It should take an optional parameter `airport` to only give back the flights that take off from there.

Help the flight crews to know where to fly to

`GET /operationsplan`, which takes a parameter `registration` and returns a list of operating instructions like this:

```
{
  "origin": "TXL",
  "destination": "MUC",
  "departure": "2018-04-13T10:00:00.000+02:00"
}
```

Overall Instructions

We would like to see some real-life code. As in the scenario, we would like to see how far you get in 3 hours. You should, however, apply all the best practices you would normally when writing code in a team setting, like tests, documentation, etc. We would like to see a git repository, either shared with us privately through github or packed into a tarball and sent via email.