Dropit

Hey, before you start, **notice** that the test is about your coding style, readability & architecture. Write it in your favourite language and try to show us what you got.

Implement a simple Delivery API containing the following:

Methods

POST **/resolve-address** - resolves a single line address into a structured one (See 'Address' model)

```
{
  "searchTerm": {SINGLE LINE ADDRESS}
}
```

POST **/timeslots** - retrieve all available timeslots (See 'Timeslot' model) for a formatted address

```
{
  "address": {FORMATTED ADDRESS}
}
```

POST **/deliveries** - book a delivery

```
{
  "user": {USER},
  "timeslotId": {TIMESLOT_ID}
}
```

POST **/deliveries/{DELIVERY_ID}/complete** - mark a delivery as completed
DELETE **/deliveries/{DELIVERY_ID}** - cancel a delivery
GET **/deliveries/daily** - retrieve all today's deliveries
GET **/deliveries/weekly** - retrieve the deliveries for current week

Models

- **Timeslot** - a delivery window containing start time, end time, supported addresses (choose by which address values to decide if its supported or not)
- **Delivery** - contains status & selected timeslot
- **Address** - should be resolved from -> https://www.geoapify.com/geocoding-api (or any other) into an object that holds at least: street, line1, line2, country, postcode

Requirements

- courier API - just a static json file with the available timeslots for the upcoming week which would be loaded on start.
- Exclude courier timeslots that fall on holidays via -> https://holidayapi.com/docs (or any other)
- The two sources above should be fetched in **parallel** in order to calculated the resulted timeslots
- Business capacity - assume that the system supports up to 10 deliveries per day.
- Each timeslot can be used for 2 deliveries
- Find a way how to handle / validate **concurrent** requests (two consumers trying to reserve the same timeslot)

Good Luck :)