Code challenge

We'd like you to submit your exercise as a Git repository so that we can see how you elaborated on your solution. Preferably, share a private **GitHub** repository with the user **s4l-techrec**. You have a week to complete it so please take your time. There are no extra points for doing it quickly. We would prefer that you take your time and deliver a solution that you're proud of.

Here are some guidelines on how to approach the problem and what to deliver:

Requirements

- Include a README.md file explaining *at least* how to install and run the application.
- Following the best practices of the programming language of your choice, your solution must show off some maintainability characteristics such as modularity, understandability, changeability, testability, and reusability.
    - *For instance, it could follow Hexagonal architecture and SOLID principles to comply with some of the above characteristics.*
- Unit testing.

Nice to have

- A Docker environment to run the application. Docker Compose would also be appreciated.
- E2E (end-to-end) testing.
- Performant for big inputs.

Programming language

- You can code your solution with any of these: Go, PHP, Java, Kotlin, Python, JavaScript, or Ruby
- If you want to use a different language, please get in touch before continuing.

Problem definition

We own an apartment and we're renting it through different popular websites. Before actually renting the apartment for some days, those platforms send us booking requests. We want to get insights from those booking requests in order to make better decisions. For instance, we'd like to know what's the profit per night we're getting and what could be the best combination of bookings to maximize our profits.

We will create an API for this purpose. This API will contain two endpoints; you can find the whole definition here:
https://app.swaggerhub.com/apis-docs/BlackfireSFL/BackendChallenge/1.0.1

/stats endpoint

Given a list of booking requests, return the *average, minimum*, and *maximum* **profit per night** taking into account all the booking requests in the payload. The concept “profit per night” follows this calculation:

*(selling_rate * margin (percentage)) / nights*

Example request:

```
POST /stats
[
    {
      "request_id":"bookata_XY123",
      "check_in":"2020-01-01",
      "nights":5,
      "selling_rate":200,
      "margin":20
    },
    {
      "request_id":"kayete_PP234",
      "check_in":"2020-01-04",
      "nights":4,
      "selling_rate":156,
      "margin":22
    }
]
```

Example response:

```
200 OK

{
  "avg_night":8.29,
  "min_night":8,
  "max_night":8.58
}
```

Example request:

```
POST /stats
[
    {
      "request_id":"bookata_XY123",
      "check_in":"2020-01-01",
      "nights":1,
      "selling_rate":50,
      "margin":20
    },
    {
      "request_id":"kayete_PP234",
      "check_in":"2020-01-04",
      "nights":1,
      "selling_rate":55,
      "margin":22
    },
    {
      "request_id":"trivoltio_ZX69",
      "check_in":"2020-01-07",
      "nights":1,
      "selling_rate":49,
      "margin":21
    }
]
```

Example response:

```
200 OK

{
  "avg_night":10.80,
  "min_night":10,
  "max_night":12.1
}
```

/maximize endpoint

Given a list of booking requests, return the best combination of requests that maximizes total profits.

Acceptance Criteria

- Two booking requests cannot overlap in time. *For instance, the following requests overlap and cannot be combined. Remember we are renting a single apartment!*
  - *A:* `check_in: 2020-01-01; nights: 5` *(check out on* `2020-01-06`*)*
  - *B:* `check_in: 2020-01-03; nights: 5` *(check out on* `2020-01-08`*)*
- If more than one combination yields the same maximum profit, return any of them.
- Response fields
  - `request_ids` - list of IDs of the best combination
  - `total_profit` - the total profit of the best combination
  - `avg_night` - the average profit per night of the best combination
  - `min_night` - the minimum profit per night of the best combination
  - `max_night` - the maximum profit per night of the best combination

Detailed example

Given this set of booking requests:

A: `check_in: 2018-01-01; nights: 10; selling_rate: 1000€; margin: 10%`
B: `check_in: 2018-01-06; nights: 10; selling_rate: 700€; margin: 10%`
C: `check_in: 2018-01-12; nights: 10; selling_rate: 400€; margin: 10%`

We will choose:

combination	valid	profit
A	YES	1000*0,1 = 100
B	YES	700*0,1 = 70
C	YES	400*0,1 = 40
A+B	NO	--
A+C	YES	1000*0,1 + 400*0,1 = 140
A+B+C	NO	--
B+C	NO	--

A+C is the best combination of bookings that do not overlap and yield the maximum profit.

With the profit being:

A.sellingRate * A.margin + C.sellingRate * C.margin = 1000 * 0,1 400 * 0,1 = 140€

The maxim profit would be: **140€** by combining **A+C**

Example request:

```
POST /maximize
[
    {
      "request_id":"bookata_XY123",
      "check_in":"2020-01-01",
      "nights":5,
      "selling_rate":200,
      "margin":20
    },
    {
      "request_id":"kayete_PP234",
      "check_in":"2020-01-04",
      "nights":4,
      "selling_rate":156,
      "margin":5
    },
    {
      "request_id":"atropote_AA930",
      "check_in":"2020-01-04",
      "nights":4,
      "selling_rate":150,
      "margin":6
    },
    {
      "request_id":"acme_AAAAA",
      "check_in":"2020-01-10",
      "nights":4,
      "selling_rate":160,
      "margin":30
    }
]
```

Example response:

```
200 OK

{
  "request_ids":[
      "bookata_XY123",
      "acme_AAAAA"
  ],
  "total_profit":88,
  "avg_night":10,
  "min_night":8,
  "max_night":12
}
```