Backend Coding Test: Availability API

Build an API that is capable of determining table availability for a restaurant given a restaurant, date, time, and number of guests. Additionally, create an API endpoint to create reservations for an available table on a specific date.

Functional requirements

Step 1: Availability request

The api must respond to a GET request containing a restaurant identifier, date/time, and number of guests and respond with a list of tables that meet the requested parameters.

Example request

```
GET
/api/availability/[restaurant]?datetime=2021-12-10T10:00:00+09:00&
guests=3
```

Example response:

200 OK

```
[
  {
    id: "xxx",
    name: "Table 1",
    qty_available: 1,
    available_from: "10:00",
    available_to: "12:00",
    min_guests: 1,
    max_guests: 4
  },
  {
    id: "xxx",
    name: "Table 2",
    qty_available: 3,
    available_from: "12:00",
    available_to: "14:00",
    min_guests: 2,
    max_guests: 4
  }
]
```

Step 2: Reservation request

Add a reservations API which will allow the user to create a reservation for a given table on a specific date. A successful request will decrement the number of tables available *for that date (keep in mind that the table may still be available for other dates).*

If the requested table is no longer available or if there is another error in the request, an appropriate error message and status code must be returned.

Example successful request

```
POST /api/reservations/[restaurant]
```

Body:

```
{
  table_id: "xxx",
  date: "2021-12-10",
  name: "Fred Jones",
  email: "fred@mysteryinc.com",
  guests: 3
}
```

Example successful response

```
204 OK
```

Additional requirements

- Code must be covered by the appropriate testing framework. 100% coverage is not required; just be sure to include unit/integration tests where you believe they provide the greatest benefit.
- Code comments and documentation must also be included where appropriate.

Tech stack

- Ruby + Rails / Elixir + Phoenix (or just Plug)
- RSpec / ExUnit
- RuboCop / Credo (optional)
- The database of your choice, eg. PostgreSQL, MySQL, MongoDB, etc.
- Git: Develop your project in a git repository that can be shared (via github, etc.).

Submitting your project

Upload your project to a publicly accessible repository and share the link with your hiring manager to be passed on for review.

Notes

- You are responsible for creating your own database schema and adding the appropriate data. You may add as many shops/table types as you would like, but be sure to include at least two shops and more than one table type to each. Feel free to add additional tables and even data sources (Redis, etc.) if you feel they would be a benefit to the project.
- It is not strictly necessary to provide seed data for your database if tests are sufficiently written, but it may be helpful (such as with “rails db:seed”, etc.)
- Try to make regular commits to your repository. Avoid submitting the whole project in a single commit.